

Universidade de Brasília - UnB
Faculdade UnB Gama - FGA
Engenharia Eletrônica

Modelagem de um PLL e Projeto de VCO para Transceptor ZigBee

Autor: Thiago Almeida Nunes Guimarães
Orientador: Prof. Dr. Wellington Avelino do Amaral

Brasília, DF
2015



Thiago Almeida Nunes Guimarães

Modelagem de um PLL e Projeto de VCO para Transceptor ZigBee

Monografia submetida ao curso de graduação em (Engenharia Eletrônica) da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em (Engenharia Eletrônica).

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Orientador: Prof. Dr. Wellington Avelino do Amaral

Brasília, DF

2015

Thiago Almeida Nunes Guimarães

Modelagem de um PLL e Projeto de VCO para Transceptor ZigBee/ Thiago
Almeida Nunes Guimarães. – Brasília, DF, 2015-
178 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Wellington Avelino do Amaral

Trabalho de Conclusão de Curso – Universidade de Brasília - UnB
Faculdade UnB Gama - FGA , 2015.

1. PLL. 2. VCO. I. Prof. Dr. Wellington Avelino do Amaral. II. Universidade
de Brasília. III. Faculdade UnB Gama. IV. Modelagem de um PLL e Projeto de
VCO para Transceptor ZigBee

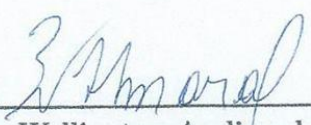
CDU 02:141:005.6

Thiago Almeida Nunes Guimarães

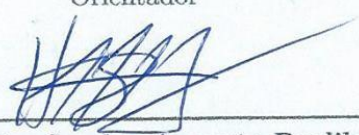
Modelagem de um PLL e Projeto de VCO para Transceptor ZigBee

Monografia submetida ao curso de graduação em (Engenharia Eletrônica) da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em (Engenharia Eletrônica).

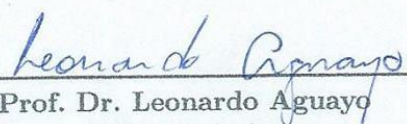
Trabalho aprovado. Brasília, DF, Julho de 2015:



Prof. Dr. Wellington Avelino do
Amaral
Orientador



Prof. Dr. Sandro Augusto Pavlik
Haddad
Convidado 1



Prof. Dr. Leonardo Aguayo
Convidado 2

Brasília, DF
2015

Este trabalho é dedicado à minha família e amigos.

Agradecimentos

Primeiramente gostaria de agradecer a minha família, que sempre me apoiou e deu toda a base necessária para que eu conseguisse permanecer na faculdade por todos esses anos. Desta forma, gostaria de agradecer aos meus pais Evando Nunes e Lúcia Vânia; minhas irmãs Luciane e Alessandra; minha sobrinha Millena; minha finada avó Zilda; minhas tias Ana Maria e Izabel; minha prima Andressa e todos os demais familiares.

Agradeço ao Prof. Dr. Wellington Avelino do Amaral pela oportunidade de desenvolver este trabalho, pela dedicada orientação e compartilhamento de conhecimento. Agradeço também ao Prof. Dr. Adson Ferreira da Rocha, a Prof.^a Dr.^a Suélia de Siqueira Rodrigues Fleury Rosa, ao Prof. Dr. Cristiano Jacques Miosso Rodrigues Mendes, ao Dr. Heider Marconi, ao Rafael Santos Ferreira, ao senhor Washington Rodrigues Póvoa e aos Profs. Dr. Sandro Augusto Pavlik Haddad e Dr. Leonardo Aguayo, estes dois últimos componentes da banca avaliadora. Todos esses, entre outros, se tornaram modelos de pessoas e profissionais os quais tentarei seguir na minha jornada como engenheiro.

E por fim, e não menos importante, gostaria de agradecer aos meus amigos, àquelas pessoas que, sem elas, esta jornada seria praticamente impossível. Portanto, gostaria de agradecer aos *roommates* Vilmei Filho, Yan Watanabe e André Mateus; aos amigos de longa data, Danilson Castelo, Kleber Granella, José Rogério, Jorge Olaff, José Pedro de Santana Neto, Leandro Borges, Thiago Coimbra, Rodrigo Reis, Geraldo Jerônimo, Igor Josafá e Francisco Júnio; aos veteranos Irani Elias, José Alberto, Jeferson Lopes, Rodrigo Calixto, Matheus Pimenta, Gilvanson Costa e Nicholas Tolentino; ao companheiro de TCC José Alisson; aos amigos do mundo da música Thiago Macedo, João Paulo Porto, Eugênio de Oliveira e Anthony Taylor; e aos amigos que fiz há pouco tempo, porém já considero bastante, Érica Costa, Beatriz Rodrigues, Hebert Douglas, Albert Almeida, João Paulo Mendonça, Marcus Vinicius Mendonça, Marlon Filho, Pedro Penaforte, Pablo Alejandro, Lívia Araújo, Stephanie Costa, Ebenezer Andrade, Amanda Gontijo, Maxwell de Oliveira, Renan Costa, Oziel da Silva e Heleno Silva.

“You were only waiting for this moment to arise.”

Paul McCartney

Resumo

A fim de estabelecer redes de comunicação sem fio (WSN) que agreguem baixo custo, baixo consumo, baixa taxa de transmissão de dados, confiabilidade e segurança, o protocolo ZigBee (IEEE 802.15.4) aparece como uma excelente alternativa, podendo ser aplicado em diversos contextos e finalidades, por exemplo em *Smart Grids*. O objetivo deste trabalho é a modelagem de um *Phase Locked Loop* (PLL) e projeto de um Oscilador Controlado por Tensão (VCO) para transceptor ZigBee. O projeto foi desenvolvido com auxílio das ferramentas Cadence seguindo a metodologia de projetos *Top-Down*. O Verilog-AMS foi a linguagem de descrição de *hardware* utilizada na modelagem, a qual possibilita a análise comportamental e simulação mista. Este trabalho foi dividido em duas partes: fundamentos teóricos (feito no TCC1) e implementação, realizada na disciplina TCC2. Na primeira parte, o levantamento inicial das especificações dos circuitos foram apresentados, resultando na proposta da topologia do PLL e em possíveis topologias para o VCO tanque LC. Na segunda parte, o sistema PLL foi completamente modelado usando Verilog-AMS e o VCO foi projetado em nível transistor, utilizando a tecnologia CMOS TSMC 0.18 μm . Os outros blocos que compõem o sistema PLL foram desenvolvidos em nível transistor por outros alunos. Em paralelo com o trabalho de modelagem, simulações mistas, utilizando os circuitos desenvolvidos neste trabalho bem como os circuitos desenvolvidos pelos outros alunos, foram realizadas mostrando resultados promissores.

Palavras-chaves: *Phase Locked Loop*. Oscilador controlado por Tensão. Transceptor. ZigBee.

Abstract

In order to establish Wireless Sensor Networks (WSN) which aggregate low cost, low power, low data rate, reliability and security, the ZigBee protocol (IEEE 802.15.4) appears as an excellent alternative and can be applied to many different contexts and proposes, e.g. Smart Grids. The objective of this work is to model a Phase Locked Loop (PLL) and designing a Voltage Controlled Oscillator (VCO) for a ZigBee transceiver. The project was developed using the Cadence tools following the Top-Down methodology for integrated circuit designs. The Verilog-AMS was the main hardware description language used in the modeling, making possible the behavioral analysis and mixed signal simulations. This academic work was divided in two parts: the theoretical foundation (done in the TCC1 discipline) and the implementation, accomplished in the TCC2 discipline. An initial survey of the circuits specifications was also done, resulting in a proposal of topologies for the PLL and the VCO LC-tank. In the TCC2 discipline, the PLL system was completely modeled using Verilog-AMS and the VCO was designed in transistor level, using the TSMC 0.18 μ m CMOS technology. The other blocks that compound the PLL system were developed in transistor level by other students. In parallel with the modeling work, mixed signal simulations, using the circuits developed in this work as well as the circuits developed by other students, were carried out showing promising results.

Key-words: Phase Locked Loop. Voltage Controlled Oscillator. Transceiver. Zig-Bee.

Lista de ilustrações

Figura 1 – Camadas do protocolo ZigBee [Silva (2008)].	32
Figura 2 – Topologias de rede [Coelho (2013)].	33
Figura 3 – Arquitetura típica de um transceptor [Ferreira (2006)].	34
Figura 4 – Aplicação do ZigBee em <i>Smart Grid</i> [Batista, Melício e Mendes (2014)].	35
Figura 5 – Etapas básicas do fluxo de projetos [Zurita (2013)].	41
Figura 6 – Ciclo <i>top-down</i> de projeto de um Circuito Integrado [Johann (1997)].	43
Figura 7 – Divisões do HDL Verilog [Melnik (2006)].	45
Figura 8 – Visão de uso do Verilog-AMS [Melnik (2006)].	46
Figura 9 – PLL básico [Bistue, Quemada e Adin (2009)].	50
Figura 10 – Diagrama de blocos do PLL [Dabhi e Nagpara (2014)].	50
Figura 11 – PFD/CP e Filtro de Malha: (a) PFD; (b) <i>Charge Pump</i> ; (c) Filtro de Malha [Henzler (2011)].	51
Figura 12 – Resposta do VCO a partir da tensão de controle [Argüello (2004)]. . . .	52
Figura 13 – Divisor de frequência <i>Pulse-Swallow</i> [Argüello (2004)].	53
Figura 14 – Arquitetura N-fracionário [Argüello (2004)].	54
Figura 15 – Diagrama usado para computar as contribuições de ruído de cada bloco [Bistue, Quemada e Adin (2009)].	55
Figura 16 – Ruído de fase típico do PLL [Bistue, Quemada e Adin (2009)].	57
Figura 17 – Ruído de fase típico da saída do PLL [Manthena (2011)].	57
Figura 18 – Modelos de osciladores: (a) Sistema linear com realimentação positiva; (b) Modelo de resistência negativa [Razavi e Behzad (1998)].	59
Figura 19 – (a) LGR do oscilador; (b) Forma de onda de saída [Berny et al. (2006)].	59
Figura 20 – Topologias de osciladores: (a) oscilador a cristal; (b) oscilador de relaxação; (c) oscilador em anel com inversores; (d) oscilador LC [Farfán (2003)].	61
Figura 21 – Resposta transiente do tanque LC ideal e não ideal [Bistue, Quemada e Adin (2009)].	63
Figura 22 – (a-c) Topologias tanque LC NMOS diferenciais [Bistue, Quemada e Adin (2009)].	64
Figura 23 – (a-c) Topologias tanque LC CMOS diferenciais [Bistue, Quemada e Adin (2009)].	65
Figura 24 – Espectro de frequência do oscilador: (a) Ideal; (b) Real [Anjos (2012)].	65

Figura 25 – Espectro de frequência do oscilador com canais adjacentes: (a) Ideal; (b) Real [Bistue, Quemada e Adin (2009)].	66
Figura 26 – Representação gráfica do modelo de Leeson [Farfán (2003)].	66
Figura 27 – Topologias de VCO propostas: (a) NMOS; (b) CMOS [Bistue, Quemada e Adin (2009)].	67
Figura 28 – Esquemático do VCO projetado.	73
Figura 29 – <i>Testbench</i> do VCO projetado.	74
Figura 30 – Simulação transiente do VCO projetado: (a) sem <i>zoom</i> ; (b) com <i>zoom</i>	75
Figura 31 – ADE da simulação do projeto elétrico do VCO.	75
Figura 32 – Simulação paramétrica para 16 valores da tensão de controle.	76
Figura 33 – Simulação PSS do VCO projetado: (a) sem <i>zoom</i> ; (b) com <i>zoom</i>	77
Figura 34 – Simulação PNOISE do VCO projetado: (a) sem <i>zoom</i> ; (b) com <i>zoom</i>	77
Figura 35 – Fluxo de projeto de um PLL [Bistue, Quemada e Adin (2009)].	79
Figura 36 – Diagrama de blocos da topologia de PLL proposta.	81
Figura 37 – Filtro de malha de segunda ordem e a sua função de transferência [Srinivasan (2006)].	82
Figura 38 – Localização dos polos e zeros [Srinivasan (2006)].	82
Figura 39 – Gráfico de Bode da resposta em malha aberta e malha fechada, respectivamente.	86
Figura 40 – Variação da resposta em malha aberta e fechada, respectivamente, com ω_n	87
Figura 41 – Variação dos componentes do filtro de malha com f_c	87
Figura 42 – Diagrama de blocos do PLL e tipos de sinais.	89
Figura 43 – Passos da Metodologia de Projeto do PLL.	90
Figura 44 – Gráfico de Bode da resposta em malha aberta e malha fechada, respectivamente.	90
Figura 45 – Esquemático do PFD_v1.	92
Figura 46 – <i>Testbench</i> do PFD.	93
Figura 47 – Simulação do PFD.	94
Figura 48 – <i>Testbench</i> do <i>charge pump</i>	95
Figura 49 – Simulação do <i>charge pump</i>	96
Figura 50 – <i>Testbench</i> do filtro de malha.	98
Figura 51 – Simulação do filtro de malha.	98
Figura 52 – <i>Testbench</i> do VCO.	100
Figura 53 – Simulação do VCO para os 16 canais do PLL.	101
Figura 54 – Simulação do VCO - Tensão de controle X Frequência.	101
Figura 55 – <i>Testbench</i> do conversor de saída diferencial para <i>single ended</i>	103

Figura 56	– Simulação do conversor de saída diferencial para <i>single ended</i> .	103
Figura 57	– Esquemático do Divisor de Frequência.	104
Figura 58	– <i>Testbench</i> do Divisor de Frequência.	105
Figura 59	– Simulação do Divisor de Frequência para os 16 canais.	106
Figura 60	– Esquemático do PLL.	107
Figura 61	– <i>Testbench</i> do PLL modelado.	108
Figura 62	– Simulação da modelagem do PLL para o canal 11 ($S < 4 \geq 0000$).	109
Figura 63	– ADE L para simulação da modelagem do PLL para o canal 11 ($S < 4 \geq 0000$).	110
Figura 64	– Simulação da modelagem do PLL para o canal 18 ($S < 4 \geq 0111$).	111
Figura 65	– ADE L para simulação da modelagem do PLL para o canal 18 ($S < 4 \geq 0111$).	112
Figura 66	– Simulação da modelagem do PLL para o canal 26 ($S < 4 \geq 1111$).	113
Figura 67	– ADE L para simulação da modelagem do PLL para o canal 26 ($S < 4 \geq 1111$).	114
Figura 68	– Sinais de saída da simulação da modelagem do PLL para o canal 18 ($S < 4 \geq 0111$).	115
Figura 69	– Resultados das equações no ADE L da simulação abaixo.	119
Figura 70	– Simulação mista do PFD, <i>charge pump</i> e filtro de malha com o modelo do PLL completo para o canal 18 ($S < 4 \geq 0111$).	120
Figura 71	– Resultados das equações no ADE L da simulação abaixo.	120
Figura 72	– Simulação mista do conversor diferencial para <i>single ended</i> com o modelo do PLL completo para o canal 18 ($S < 4 \geq 0111$).	121
Figura 73	– Resultados das equações no ADE L da simulação abaixo.	121
Figura 74	– Simulação mista entre os divisores de frequência modelado e projetado.	122
Figura 75	– Erro entre os divisores de frequência modelado e projetado.	122
Figura 76	– Esquemático do <i>Main Counter</i> .	133
Figura 77	– <i>Testbench</i> do <i>Main Counter</i> .	133
Figura 78	– Simulação do <i>Main Counter</i> .	134
Figura 79	– Esquemático do <i>Prescaler</i> .	135
Figura 80	– <i>Testbench</i> do <i>Prescaler</i> .	135
Figura 81	– Simulação do <i>Prescaler</i> .	136
Figura 82	– Fatores de divisão do <i>Prescaler</i> : (a) 15; (b) 16.	136
Figura 83	– Esquemático do <i>Scounter</i> .	137
Figura 84	– <i>Testbench</i> do <i>Scounter</i> .	137
Figura 85	– Simulação do <i>Scounter</i> .	138
Figura 86	– Configuração do editor de texto usado para desenvolver código em Verilog-AMS.	157

Figura 87	– Criação da <i>library</i> para desenvolvimento do projeto.	157
Figura 88	– Criação das <i>cellviews</i> necessárias para desenvolvimento do projeto: (a) Vista <i>verilogams</i> ; (b) Vista <i>schematic</i> ; (c) Vista <i>config</i>	157
Figura 89	– <i>Testbench</i> do conversor AD de 16 <i>bits</i> modelado em Verilog-AMS, vista <i>schematic</i>	158
Figura 90	– Mensagens após compilação do código: (a) <i>Errors/Warnings</i> ; (b) Criação do símbolo.	159
Figura 91	– <i>Template</i> usado na vista <i>config</i> do conversor AD.	159
Figura 92	– Vista <i>config</i> do conversor AD.	159
Figura 93	– Abertura da vista <i>config</i>	160
Figura 94	– Escolha do simulador AMS para simulações no ADE L.	160
Figura 95	– ADE L para simulação do conversor AD modelado.	160
Figura 96	– Parâmetros disponíveis para o conversor AD modelado.	161
Figura 97	– Simulação do conversor AD de 16 <i>bits</i> modelado em Verilog-AMS. . . .	161
Figura 98	– Resultados obtidos do código D.1.	173
Figura 99	– Resultados obtidos do código D.1 com modificação do ganho do VCO. .	174
Figura 100	– Esquemático do fonte de corrente usada no VCO.	177
Figura 101	– <i>Testebench</i> da fonte de corrente usada no VCO.	178
Figura 102	– Simulação da fonte de corrente usada no VCO.	178

Lista de tabelas

Tabela 1 – Bandas de frequência e taxa de dados [Ergen (2004)].	32
Tabela 2 – Funções de transferência das fontes de ruído do PLL [Banerjee (2006)].	56
Tabela 3 – Especificações 2450 MHz IEEE 802.15.4 camada PHY [Oh e Lee (2006)].	80
Tabela 4 – Parâmetros do filtro de malha com a variação de ω_n	88
Tabela 5 – Parâmetros de simulação comuns aos blocos.	91
Tabela 6 – Descrição dos pinos do detector de fase e frequência.	93
Tabela 7 – Parâmetros de simulação do detector de fase e frequência.	93
Tabela 8 – Descrição dos pinos do <i>charge pump</i>	95
Tabela 9 – Parâmetros de simulação do <i>charge pump</i>	96
Tabela 10 – Descrição dos pinos do filtro de malha.	97
Tabela 11 – Parâmetros de simulação do filtro de malha.	97
Tabela 12 – Descrição dos pinos do VCO.	100
Tabela 13 – Parâmetros de simulação do VCO.	100
Tabela 14 – Descrição dos pinos do conversor de saída diferencial para <i>single ended</i> .	102
Tabela 15 – Parâmetros de simulação do conversor de saída diferencial para <i>single ended</i>	103
Tabela 16 – Descrição dos pinos do Divisor de Frequência.	105
Tabela 17 – Parâmetros de simulação do Divisor de Frequência.	105
Tabela 18 – Descrição dos pinos do PLL completo.	108
Tabela 19 – Parâmetros de simulação do PLL completo.	115
Tabela 20 – Frequência de Saída do PLL modelado.	117
Tabela 21 – Fator de Divisão do PLL modelado.	117
Tabela 22 – Tensão de Controle do VCO.	118
Tabela 23 – <i>Settling Time</i> do PLL modelado.	118
Tabela 24 – Descrição dos pinos do <i>Main Counter</i>	134
Tabela 25 – Descrição dos pinos do <i>Prescaler</i>	136
Tabela 26 – Descrição dos pinos do <i>Scounter</i>	138

Lista de abreviaturas e siglas

ADS	<i>Advanced Design System</i>
AMS	<i>Analog Mixed-Signal</i>
APL	<i>Application</i>
BIAS	Polarização do circuito
BPSK	<i>Binary Phase Shift Keying</i>
BW	<i>Bandwidth</i>
CI	Circuito Integrado
CMOS	<i>Complementary Metal-Oxide-Semiconductor</i>
CP	<i>Charge Pump</i>
DC	<i>Direct Current</i>
DSSS	<i>Direct Sequence Spread Spectrum</i>
FFD	<i>Full Function Devices</i>
FM	<i>Frequency Modulation</i>
GND	<i>Ground</i>
HDL	<i>Hardware Description Language</i>
IEEE	<i>Institute of Electrical and Electronics Engineers</i>
IF	<i>Intermediate Frequency</i>
IP	<i>Intellectual Property</i>
ISM	<i>Industrial, Scientific and Medical</i>
LF	<i>Loop Filter</i>
LO	<i>Local Oscillator</i>
MAC	<i>Medium Access Control</i>
NMOS	Transistor MOS de Canal N

NWK	<i>Network</i>
OQPSK	<i>Offset Quadrature Phase-Shift Keying</i>
PD	<i>Phase Detector</i>
PFD	<i>Phase Frequency Detector</i>
PHY	<i>Physical</i>
PLL	<i>Phase Locked Loop</i>
PMOS	Transistor MOS de Canal P
PNOISE	<i>Periodic Noise</i>
PSS	<i>Periodic Stady State</i>
RF	<i>Radio Frequency</i>
RFD	<i>Reduced Funcion Devices</i>
SNR	<i>Signal to Noise Ratio</i>
SSB	<i>Single Side Band</i>
TCC	Trabalho de Conslusão de Curso
TCXO	<i>Temperature Compensated Crystal Oscillator</i>
TF	<i>Transfer Function</i>
TSMC	<i>Taiwan Semiconductor Manufacturing Company</i>
VCO	<i>Voltage Controlled Oscillator</i>
VDD	<i>Supply Voltage</i>
Verilog	VERIfying LOGic
VHDL	<i>VHSIC Hardware Description Language</i>
VHSIC	<i>Very High Scale Integrated Circuits</i>
VLSI	<i>Very Large Scale Integration</i>
WAMR	<i>Wireless Automatic Meter Reading</i>
WPAN	<i>Wireless Personal Area Network</i>
WSN	<i>Wireless Sensor Network</i>

Lista de símbolos

α	Precisão da frequência
$A(t)$	Amplitude do VCO no tempo
agnd	Pino de <i>ground</i> analógico
ain	Entrada analógica
ampl	Parâmetro de amplitude do sinal
avdd	Pino de referencial de tensão analógico
C	Capacitância
C_1	Capacitância 1 do Filtro de Malha
C_2	Capacitância 2 do Filtro de Malha
ciclos	Quantidade de ciclos de simulação
Δf	Faixa de frequência máxima de sintonização
$\Delta\omega$	Deslocamento de frequência em relação a portadora
dout	Saída digital
DOWN	Sinal de saída do PFD
en	<i>Enable</i>
f	Frequência
f_{fr}	Frequência de funcionamento
f_{osc}	Frequência de oscilação do VCO
f_0	Frequência da portadora
F	Fator empírico que leva em conta o aumento da densidade de ruído na região $(1/\Delta\omega)^2$, e ω_1/f_3 é a frequência que limita as regiões $(1/\Delta\omega)^2$ e $(1/\Delta\omega)^3$ do modelo de Leeson
f_{CANAL}	Frequência de sintonização dos canais
f_{div}	Frequência de saída divisor

f_{max}	Frequência máxima do VCO
f_{min}	Frequência mínima do VCO
f_{ref}	Frequência de referência
f_{VCO}	Frequência de saída do VCO
f_{VCOc}	Frequência de saída central do VCO
$F(s)$	Ganho de malha fechada do PLL
$G(s)$	Ganho de malha aberta do PLL
$H(s)$	Função de transferência do divisor
I_{CP}	Corrente do <i>Charge Pump</i>
I_{PFD}	Ruído de fase do PFD
I_{VCO}	Ruído de fase do VCO
I_{XTAL}	Ruído de fase da referência
k	Constante de Boltzmann
L	Indutância
M	<i>Dual Modulus</i>
N	Fator de divisão
P	<i>Programmable Counter</i>
P_{BW}	Potência do conteúdo do sinal por toda a largura do canal
$P_{carrier}$	Potência da portadora
P_{int}	Potência do conteúdo da interferência
P_{LO}	Potência do conteúdo do LO
PM	<i>Phase Margin</i>
PN	Contribuição do LO ao ruído de fase
P_{sig}	Potência do conteúdo da portadora
P_{sp}	Contribuição do LO para emissão de sinais espúrios
Q	Fator de qualidade do tanque

R_1	Resistência 1 do Filtro de Malha
S	<i>Swallow counter</i>
$S < 4 >$	Palavra binária para seleção do canal do PLL
SNR_{MIN}	Relação sinal ruído mínima na entrada da seção IF
t_{lock}	<i>Settling time</i>
T	Temperatura absoluta
t_d	Tempo de <i>delay</i>
t_f	Tempo de transição
$T(s)$	Função de transferência da tensão ruído das resistências do Filtro de Malha
$\theta(t)$	Fase do VCO
UP	Sinal de saída do PFD
V_{in}	Tensão de entrada
$V_{in_dc_{max}}$	Tensão DC máxima de entrada do VCO
$V_{in_dc_{min}}$	Tensão DC mínima de entrada do VCO
V_{th}	Tensão de <i>threshold</i>
V_{ctrl}	Tensão de controle do VCO
ω	Frequência de <i>offset</i> com respeito a frequência de entrada do PLL
ω_0	Frequência de oscilação
ω_c	Largura de banda do PLL
ω_n	Frequência natural do PLL
ω_{p1}	Frequência do polo 1
ω_{z1}	Frequência do zero 1
Out_N	Saída diferencial N
Out_P	Saída diferencial P
ζ	Fator de amortecimento do PLL
$Z(s)$	Função de transferência do Filtro de Malha

Sumário

1	Introdução	31
1.1	Aspectos Gerais	31
1.1.1	Características da Rede ZigBee	31
1.2	Objetivos	33
1.3	Motivação: Aplicações em <i>Smart Grid</i>	34
1.3.1	Leitura Automática Sem Fio	35
1.3.2	Falhas de Linhas de Transmissão e Detecção de Furto de Energia	35
1.4	Organização do Trabalho	36
I	Revisão Bibliográfica	39
2	Metodologias de Projeto	41
2.1	Metodologia <i>Top-Down</i>	42
3	Verilog	45
3.1	Verilog-AMS	45
3.1.1	Características	46
4	PLL	49
4.1	Funcionamento do PLL	49
4.2	Componentes do PLL	50
4.2.1	PD/PFD	51
4.2.2	<i>Charge Pump</i> (CP)	51
4.2.3	Filtro de Malha	51
4.2.4	VCO	52
4.2.5	Divisor	52
4.3	Parâmetros do PLL	52
4.4	Tipos de Arquitetura	53
4.4.1	N-Inteiro	53
4.4.2	N-Fracionário	54
4.5	Ruído de Fase	55
4.6	Emissão de Espúrios	57
4.7	Proposta de Topologia de PLL para ZigBee	58
5	VCO	59
5.1	Parâmetros do VCO	60

5.2	Tipos de VCO	61
5.2.1	Cristal	61
5.2.2	Relaxação	61
5.2.3	Em anel	62
5.2.4	Tanque LC	62
5.2.4.1	Tipos de VCO tanque LC	63
5.3	Ruído de Fase no VCO	65
5.4	Proposta de Topologia do VCO	67

II Projeto, Implementação e Resultados 69

6	Projeto do VCO	71
6.1	Especificações para projeto do VCO	71
6.2	Procedimento de projeto do VCO	72
6.3	Resultados do projeto elétrico do VCO	73

7	Projeto do PLL	79
7.1	Especificações do Sintetizador	80
7.2	Projeto a Nível de Sistema	81
7.3	Procedimento de Projeto	82

8	Planejamento da Modelagem do PLL	89
----------	---	-----------

9	Modelagem do PLL	91
9.1	Detector de Fase e Frequência	91
9.1.1	Descrição do Bloco	92
9.1.2	Descrição dos Pinos	93
9.1.3	Simulação	93
9.2	<i>Charge Pump</i>	94
9.2.1	Descrição do Bloco	94
9.2.2	Descrição dos Pinos	95
9.2.3	Simulação	95
9.3	Filtro de Malha	96
9.3.1	Descrição do Bloco	96
9.3.2	Descrição dos Pinos	97
9.3.3	Simulação	97
9.4	VCO	99
9.4.1	Descrição do Bloco	99
9.4.2	Descrição dos Pinos	100
9.4.3	Simulação	100

9.5	Conversor de Saída Diferencial para <i>Single Ended</i>	102
9.5.1	Descrição do Bloco	102
9.5.2	Descrição dos Pinos	102
9.5.3	Simulação	103
9.6	Divisor de Frequência	104
9.6.1	Descrição do Bloco	104
9.6.2	Descrição dos Pinos	105
9.6.3	Simulação	105
9.7	PLL completo	106
9.7.1	Descrição do Bloco	106
9.7.2	Descrição dos Pinos	108
9.7.3	Simulação	108
9.7.3.1	Equações	116
9.7.3.2	Tabelas	116
10	Simulações Mistas	119
10.1	PFD, <i>Charge Pump</i> e Filtro de Malha	119
10.2	Conversor Diferencial para <i>Single Ended</i>	120
10.3	Divisor de Frequências e Conversor Diferencial para <i>Single Ended</i>	121
III	Conclusão	123
11	Conclusão	125
11.1	Trabalhos Futuros	126
	Referências	127
	Apêndices	131
	APÊNDICE A Simulações Adicionais	133
A.1	Divisor de Frequências	133
	APÊNDICE B Códigos da Modelagem em Verilog-AMS	139
B.1	Detector de Fase e Frequência (PFD)	139
B.2	<i>Charge Pump</i> (CP)	140
B.3	Filtro de Malha	141
B.4	Oscilador Controlado por Tensão (VCO)	142
B.5	Conversor de Saída Diferencial para <i>Single Ended</i>	143
B.6	Portas Lógicas	145
B.7	Multiplexador	153

B.8	<i>Flip-Flops</i>	154
APÊNDICE C	Exemplo de Modelagem em Verilog-AMS utilizando ferramentas Cadence	157
APÊNDICE D	Código em MATLAB	163
D.1	Função para gerar parâmetros do PLL	163
Anexos		175
ANEXO A	Fonte de Corrente	177

1 Introdução

1.1 Aspectos Gerais

Atualmente, existem diversos protocolos para suporte de comunicação sem fio, entre estes encontram-se o *Bluetooth* e o *Wi-fi*. Entretanto, visando atender às necessidades de redes sem fio de curto alcance e baixa taxa de tráfego de dados voltadas para aplicações de monitoramento e controle, especialmente com o uso de baterias, foi introduzido em 2004 pela *ZigBee Alliance*, o padrão ZigBee, especificado pela norma IEEE 802.15.4.

O ZigBee é um novo padrão para redes de telemetria sem fio, otimizadas para baixo consumo e um longo período de operação da bateria. A pilha protocolar ZigBee tem suporte para redes auto-organizáveis de dispositivos nas topologias árvore, malha e estrela, permitindo a instalação rápida de um sistema de telemetria sem fio [Norris (2005)].

De modo geral, o ZigBee pode ser visto como uma adaptação do *Wi-fi* para redes do tipo WPAN (*Wireless Personal Area Network*), visando aplicações em redes de sensores sem fio, caracterizando-se como um tipo de sistema WSN (*Wireless Sensor Network*). Desta forma, o padrão ZigBee tem como principais objetivos o baixo custo, baixo consumo, baixa latência, possibilidade de implementação de redes com elevado número de dispositivos, baixa complexidade e alta durabilidade da bateria dos dispositivos, tudo isto com confiabilidade e segurança na rede.

As características expostas anteriormente implicam em diversas aplicações que podem ser beneficiadas com o uso deste protocolo, principalmente quando se necessita de baixo custo, baixo consumo, longa duração das baterias e possa pagar o preço de uma baixa taxa de transferência de dados. Algumas destas aplicações são relacionadas a automação comercial e residencial, cuidado médico pessoal, controle industrial, monitoramento ambiental, agricultura e *smart grids*, no qual é possível utilizar este tipo de rede para monitoramento e controle de diversas grandezas para as mais variadas situações, desde monitoramento de pacientes até manutenção da rede elétrica. A aplicação em redes inteligentes (*smart grids*) será melhor abordada posteriormente.

1.1.1 Características da Rede ZigBee

Assim como o *Wi-fi* e o *Bluetooth*, o ZigBee utiliza a banda de rádio ISM (Industrial, Scientific and Medical), no qual não é necessário licença para uso. A Tabela (1), ilustra algumas das características das 3 bandas utilizadas pelo ZigBee, na Europa, Estados Unidos e global, respectivamente. No âmbito global, atua-se na frequência 2.4GHz, com taxa de transmissão de 250Kbps e 16 canais disponíveis.

Tabela 1 – Bandas de frequência e taxa de dados [Ergen (2004)].

PHY(MHz)	Faixa de Frequência (MHz)	Região	Taxa de transferência (kbps)	Quantidade de canais disponíveis
868	868-868.8	Europa	20	1
915	902-928	Estados Unidos	40	10
2450	2400-2483.5	Global	250	16

A pilha de protocolar ZigBee é formada por camadas, sendo estas, a camada física (PHY), camada de acesso ao meio (MAC), camada de rede (NWK) e camada de aplicação (APL), seguindo a ordem apresentada na Fig.(1).



Figura 1 – Camadas do protocolo ZigBee [Silva (2008)].

As camadas PHY e MAC são especificadas conforme o padrão IEEE 802.15.4. PHY é a camada responsável por permitir a transmissão dos PDUs (*Protocol Data Units*) e reportar canais livres. Já a camada MAC é responsável pelo processo do encapsulamento dos dados vindo das camadas superiores, preparando-os para serem transmitidos. As duas camadas superiores, NWK e APL são administradas pelo protocolo ZigBee, de forma que a camada NWK controla a estrutura de rede, cuida do roteamento e das funções de segurança das mensagens transmitidas, enquanto a camada APL carrega o código de cada aplicação, Monsignore (2007).

Há dois tipos de classificação para os dispositivos ZigBee, os FFD (*Full Function Devices*) e os RFD (*Reduced Function Devices*). Os dispositivos FFD ou dispositivos de funções completas são empregados como coordenadores da rede, tendo acesso a todos

os outros dispositivos. Já os RFD ou dispositivos de funções reduzidas são utilizados como dispositivos terminais, com funções limitadas e que podem se portar apenas aos coordenadores. Portanto, os FFD são mais complexos, gastam mais energia e necessitam de um hardware mais potente para implantação da pilha de protocolos, enquanto os RFD são mais simples, consomem menos e são implementados com quantidade mínima de recursos possíveis de hardware.

Os dispositivos podem assumir 3 papéis numa rede, atuando como coordenadores, roteadores e terminais. Os coordenadores são dispositivos FFD e possuem papel de inicialização, distribuição de endereços, manutenção da rede, reconhecimento dos nós, entre outros. Os roteadores, também configurados como dispositivos FFD, são responsáveis pelo encaminhamento das mensagens entre os nós da rede, podendo expandi-la. Por fim, os dispositivos terminais, RFD, tem como função hospedar os sensores e atuadores.

As redes possuem topologias que são utilizadas para determinadas aplicações, podendo ser mais robusta, econômica, centralizadora ou distribuída. As topologias estão apresentadas na Fig.(2), estas são: árvore, estrela ou malha.

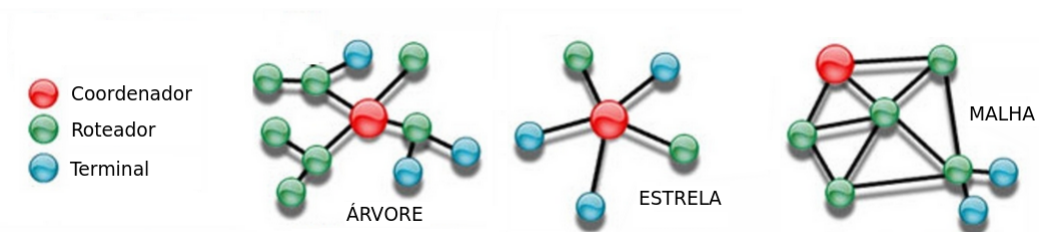


Figura 2 – Topologias de rede [Coelho (2013)].

1.2 Objetivos

Este trabalho é parte integrante de um projeto maior, composto por 3 alunos, que possui objetivo de modelar e prototipar um PLL para transceptor ZigBee, Fig.(3). Dentro do escopo deste projeto, o objetivo geral deste trabalho é a modelagem de um PLL (*Phase Locked Loop*) e projeto de um VCO (*Voltage Controlled Oscillator*). Como metodologia de projeto será utilizada a metodologia *Top-Down* com auxílio da linguagem descrição de *hardware* Verilog-AMS, o que possibilita o acompanhamento cuidadoso de todas as etapas de projeto e também a simulação de sinais mistos.

A modelagem do PLL consiste em descrever cada bloco que compõe o mesmo em linguagem de alto nível, por meio da HDL Verilog-AMS. Os blocos a serem descritos são: Detector de Fase e Frequência, *Charge Pump*, Filtro de Malha, VCO, Divisor de Frequências e Conversor Diferencial para *Single Ended*. O projeto do bloco VCO, por sua vez, será desenvolvido em baixo nível de abstração, ou seja, a nível de transistor, utilizando a tecnologia TSMC 0.18 μ m. Ambas as etapas serão desenvolvidas através do fluxo de projeto das ferramentas Cadence e adotando a metodologia *Top-Down*.

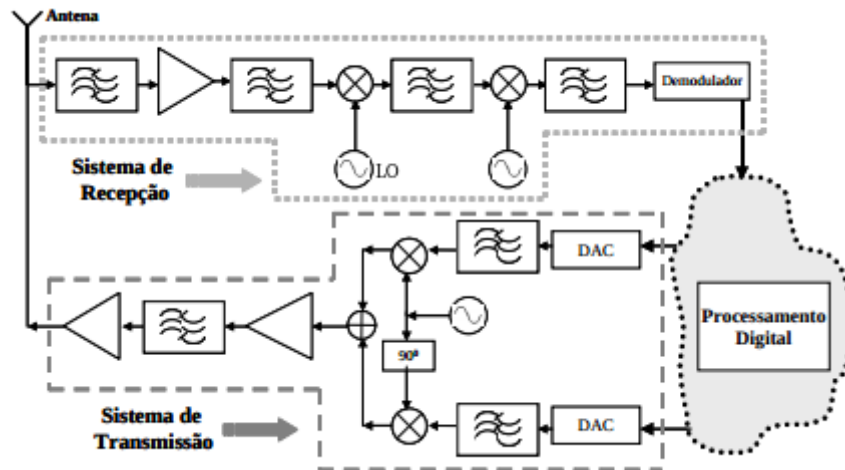


Figura 3 – Arquitetura típica de um transceptor [Ferreira (2006)].

A Fig.(3) ilustra a arquitetura típica de um transceptor. Transceptor é um dispositivo formado pela junção de dois sistemas, recepção e transmissão, onde a maioria dos circuitos presentes é comum, tanto para transmitir como para receber. Podem ser classificados como *full-duplex* ou *half-duplex*, no qual o primeiro apresenta as funções de recepção e transmissão simultaneamente e o segundo em momentos distintos. Os componentes básicos são: amplificadores, filtros, osciladores e misturadores.

1.3 Motivação: Aplicações em *Smart Grid*

Com a integração de tecnologias de redes inteligentes (*smart grids*) com a rede de elétrica, espera-se um aumento significativo na confiabilidade e segurança do sistema energético, assim como, simultaneamente, um maior contato do usuário final com as tomadas de decisões sobre seu consumo de energia. O principal benefício da *smart grid* é a integração de sistemas de controle e monitoramento inteligentes e de baixo custo com a rede elétrica, possibilitando uma comunicação bidirecional entre os componentes do sistema elétrico. As principais aplicações de rede de energia elétrica inteligentes incluem a medição automática, monitoramento do sistema de energia e controle remoto, detecção de fraude de energia elétrica, diagnósticos de falhas, resposta à demanda, controle de carga e automação da distribuição [Bilgin e Gungor (2012)].

A maioria dos sistemas de controle e monitoramento utilizam comunicação cabeada, entretanto, esta solução tem se tornado inviável devido ao alto custo de instalação e manutenção. Os sistemas de controle e monitoramento *on-line* estão ganhando espaço devido aos avanços das WSNs, uma vez que estas estão se tornando cada vez mais seguras e confiáveis. Estes sistemas, podem monitorar dados importantes como tensão, corrente, temperatura, e outros dados relacionados, transmitindo-os para uma central ou realizando o processamento dos dados localmente num sistema de processamento de dados presente

nos nós, instalados nos equipamentos críticos da *smart grid*. [Bilgin e Gungor (2012)]. Portanto, WSNs são de grande importância na criação de redes elétricas inteligentes e de alta confiabilidade. Algumas aplicações de WSN, usando ZigBee, em *smart grids* serão apresentadas a seguir:

1.3.1 Leitura Automática Sem Fio

Com os sistemas WAMR (*Wireless Automatic Meter Reading*), o consumo de energia dos usuários podem ser coletados *on-line* e, assim, automatizar o processo de leitura do totalizador eletromecânico, reduzindo os custos operacionais dos serviços públicos [Bilgin e Gungor (2012)]. Estes sistemas podem ser implementados com WSNs garantindo assim baixo custo e baixo consumo para a comunicação sem fio.

1.3.2 Falhas de Linhas de Transmissão e Detecção de Furto de Energia

Os apagões, problemas de qualidade e furtos de energia geram grandes despesas as empresas do setor. Os principais motivos para estes problemas são a falta de monitoramento *on-line*, descoordenação de dispositivos de proteção, adulteração do medidor, irregularidades de cobrança e ligações clandestinas. Utilizando redes de sensores em equipamentos essenciais, pode-se fornecer o monitoramento *on-line* da rede elétrica de forma a evitar ou minimizar grande parte destes problemas [Devidas e Ramesh (2010)].

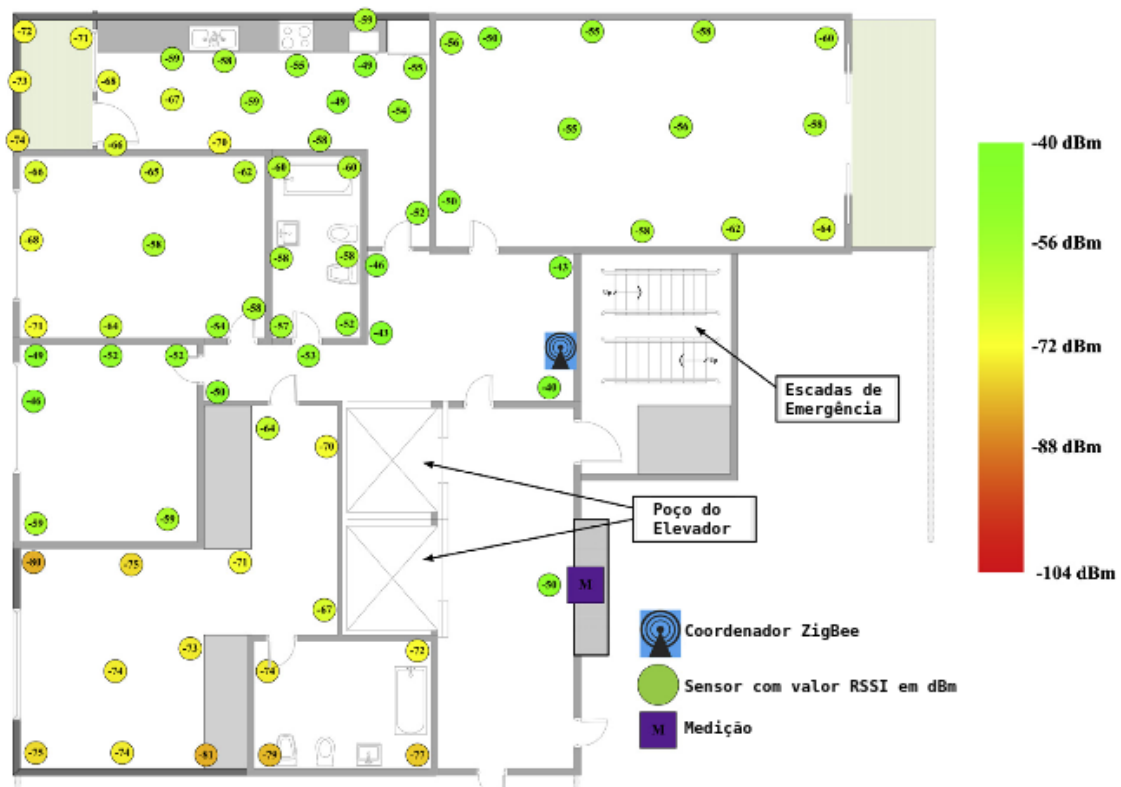


Figura 4 – Aplicação do ZigBee em *Smart Grid* [Batista, Melício e Mendes (2014)].

A Fig.(4) representa um sistema de monitoramento de fornecimento de energia sobre o comportamento do consumo. Observa-se que foi usado uma WSN, composta por um dispositivo coordenador ZigBee e diversos sensores. O monitoramento é feito a partir das leituras dos dispositivos terminais (sensores) que se comunicam com o sistema de medição e o coordenador ZigBee, este último é posicionado próximo a um ponto de acesso a *internet* para que possa haver a comunicação deste com uma central para colhimento dos dados. A partir da disponibilidade destes dados para o usuário, pode-se tomar atitudes frente ao fornecimento de energia de acordo com o perfil de consumo de energia do local.

1.4 Organização do Trabalho

Para melhor entendimento, este documento é dividido em duas 3 partes: Revisão Bibliográfica; Projeto, Implementação e Resultados; e Conclusão. A parte I é referente aos fundamentos teóricos necessários para entendimento do projeto, onde cada capítulo aborda um tema essencial. A parte II apresenta todo o procedimento de implementação e resultados, sendo composta pela modelagem do PLL, projeto elétrico do VCO e simulações mistas. A parte III é a conclusão, onde são expostas considerações sobre o projeto e possíveis trabalhos futuros.

Capítulo 1 - Introduz os componentes fundamentais do texto, objetivos, motivação e os aspectos básicos da tecnologia a qual será usada como aplicação.

Capítulo 2 - Este Capítulo trata dos aspectos fundamentais da metodologia de projeto *Top-Down*, sendo esta a metodologia de projeto que será usada.

Capítulo 3 - Apresenta uma breve descrição da HDL Verilog-AMS, linguagem que será usada para modelagem do sistema em alto nível. O Capítulo introduz as características, composição do código e descrição de algumas funções.

Capítulo 4 - São abordados temas sobre o PLL, tais como, funcionamento, componentes, características, tipos, arquiteturas, parâmetros, figuras de mérito e a proposta da topologia que será usada para a aplicação no ZigBee.

Capítulo 5 - Introduz os conceitos básicos de funcionamento do VCO, parâmetros importantes, os tipos de VCO e as topologias do tipo de interesse para a aplicação. Ao fim do capítulo, propõe-se as topologias mais adequadas à aplicação.

Capítulo 6 - Aborda o procedimento de projeto elétrico do VCO tanque LC e seus resultados.

Capítulo 7 - É desenvolvido o processo de modelagem a nível de sistema do PLL, levando em consideração a topologia proposta anteriormente e parâmetros fundamentais do circuito.

Capítulo 8 - Mostra as considerações iniciais para a modelagem do PLL em alto nível, tais como diagrama de blocos completo e tipos de sinais esperados para cada bloco.

Capítulo 9 - Implementação da modelagem do PLL completo e resultados, apresenta-se o procedimento de modelagem de cada bloco, incluindo breve descrição do bloco, pinagem, esquemático, *testbench* e simulação.

Capítulo 10 - Mostra os resultados das simulações mistas, validando o funcionamento dos projetos elétricos de cada bloco perante a modelagem do PLL completo em alto nível.

Capítulo 11 - Expõe as considerações finais do projeto e apresenta uma breve descrição dos trabalhos futuros para continuidade do trabalho.

Apêndice A - Exibe simulações adicionais do projeto, blocos internos do Divisor de Frequência.

Apêndice B - Apresenta os códigos completos de cada bloco da modelagem em Verilog-AMS.

Apêndice C - Expõe um exemplo de modelagem em Verilog-AMS utilizando ferramentas Cadence.

Apêndice D - Exibe o código em MATLAB gerado para calcular os parâmetros de interesse da modelagem do PLL a nível de sistema.

Anexo A - Apresenta breve descrição, esquemático, *testbench* e simulação da fonte de corrente usada no projeto do VCO.

Parte I

Revisão Bibliográfica

2 Metodologias de Projeto

A adequação ao ambiente de trabalho necessita de um processo de formação da estratégia onde exista participação de todos os níveis hierárquicos [Nonaka (1988)]. Metodologias de projeto são um conjunto de técnicas a fim de sistematizar as etapas de fluxo de projeto. Basicamente, são formas de iniciar, planejar e executar projetos. Desta forma, as metodologias são importantes para o êxito dos mesmos, assegurando o cumprimento das etapas, facilitando a comunicação entre os integrantes e formalizando o avanço do projeto como um todo. As etapas básicas para fluxos de projeto estão descritas a seguir (Fig. 5):



Figura 5 – Etapas básicas do fluxo de projetos [Zurita (2013)].

A etapa de especificação e modelagem consiste em especificação funcional, no levantamento de propriedades a serem satisfeitas, em índices de desempenho e nas restrições consideradas a partir dos índices de desempenho. Já a etapa de validação, visa assegurar a qualidade do produto, determinando, durante o ciclo de vida do projeto, a confiabilidade dos requisitos. Por fim, a etapa de síntese tem finalidade de transformar as especificações abstratas em menos abstratas, ou seja, moldá-las com mais detalhamento [Zurita (2013)].

No contexto deste trabalho, as abordagens mais interessantes de fluxo de projetos a serem expostas são: *Botton-up*, *Top-down* e *Middle-out*. Estas são abordagens de processamento de informação e ordenação do conhecimento quanto se trata do processo de formação da estratégia.

Para Kundert e Chang (2005), a abordagem *botton-up* se inicia de forma ascendente, do baixo para o alto nível de abstração, com o projeto e verificação de blocos individuais, a partir de determinadas especificações, onde a combinação destes blocos formam subsistemas, que por sua vez são conectados para formarem subsistemas maiores e assim sucessivamente, até estabelecer o sistema completo.

Top-down é a abordagem que descreve o sentido inverso da metodologia *botton-up*, ou seja, esta é descendente, partindo da especificação do sistema, em nível alto de abstração, para as características finais, com baixo nível de abstração [Zurita (2013)].

Portanto, a medida que o projeto avança, o sistema é refinado e subdividido, resultando numa descrição detalhada dos componentes elementares.

A metodologia *middle-out*, pode ser vista como uma combinação das anteriores, de forma que o projeto é iniciado com nível intermediário de abstração e a partir destes dados, avançar para ambos os sentidos, obtendo descrição mais refinada (*top-down*) ou compondo subsistemas mais complexos a partir daqueles descritos a nível intermediário (*botton-up*) [Zurita (2013)].

2.1 Metodologia *Top-Down*

Como explicado acima, a metodologia *top-down*, parte de um nível mais abrangente para um nível mais específico, com foco em performance. Kundert e Chang (2005) dizem que a metodologia *top-down* de projeto e verificação procede sistematicamente de arquitetura para o projeto até o nível de transistor. Cada passo é totalmente verificado antes de prosseguir para o subsequente. Um processo de verificação *top-down* também formaliza e melhora a comunicação entre projetistas, reduzindo o número de falhas devido a falta de comunicação, mesmo que eles estejam em setores distintos. Os princípios básicos no qual um sistema *top-down* efetivo é baseado são:

- Representação compartilhada do projeto entre todos os membros. Permite que o projeto seja simulado por todos os membros da equipe e em todos os tipos de descrições (comportamental, circuito, leiaute), pode ser também co-simulado, inclusive em níveis mistos. Uma vez que todas as partes integrantes do projeto estão trabalhando no mesmo ambiente, a comunicação é otimizada e potenciais problemas podem ser tratados em grupo;
- Verificação contextual completa, quando há mudanças no projeto. Durante o processo de projeto, cada alteração é verificada no contexto geral, como ditado pelo plano de verificação do projeto, garantindo que tais modificações não fujam das especificações iniciais do sistema, desta forma, mudanças parciais de circuitos geram mínimo impacto no final do fluxo de projeto;
- Um procedimento de projeto que inclui o planejamento cuidadoso na verificação de cada etapa. Este princípio é importante para que o fluxo de projeto seja bem desenvolvido, antecipando e prevenindo problemas durante as montagens dos blocos, tanto na implementação em alto nível quanto em baixo nível, ou seja, esta verificação permite uma reação a mudanças tardias, tornando-se possível atender com mais urgência áreas problemáticas da equipe;
- Múltiplos passos durante a execução, começando de alto nível de abstração e gerando refinamentos quando detalhes são disponibilizados, detalhes formulados através de

estimativas de representação de *desing* geradas por fontes confiáveis. Esta abordagem é uma medida para minimizar problemas imprevistos durante a execução do projeto, expondo possíveis deficiências desde o início, reduzindo cenários onde é preciso alterações a nível de arquitetura e especificações;

- Uso de *script* e modelos executáveis. Sempre que possível, as especificações e os planos devem ser colocados como modelos executáveis e *scripts*, evitando erros de projeto devido a falhas de comunicação, pois estes modelos são específicos e seu uso pode eliminar certas ambiguidades em descrições textuais.

Segundo [Johann \(1997\)](#), para caracterizar o ciclo de projeto de forma simplificada, adota-se um modelo sequencial de projeto *top-down*. Um exemplo é apresentado a seguir.

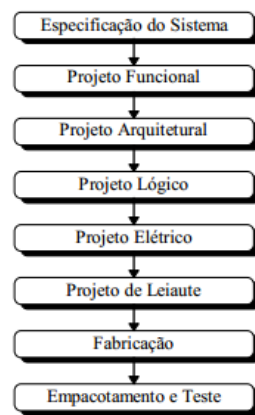


Figura 6 – Ciclo *top-down* de projeto de um Circuito Integrado [[Johann \(1997\)](#)].

A especificação do sistema consiste em detalhar o mesmo em suas interfaces, protocolos, opções de arquitetura e desempenho, levando ao projeto funcional, que por sua vez tem como finalidade obter a descrição comportamental abstrata do sistema que funcione de acordo com as especificações, sendo interessante o uso de HDLs. Segundo o comportamento do sistema, escolhe-se as opções arquiteturais para subdivisões e implementação interna de cada subdivisão, etapa denominada de projeto arquitetural. Posteriormente, no projeto lógico, o sistema é refinado estruturalmente, no qual módulos definidos na arquitetura são detalhados. Caso elementos destes módulos não possuam descrição em bibliotecas fixas, é realizado seu projeto elétrico a partir de elementos básicos, nesta etapa, a nível do sistema, é considerado seu funcionamento e coerência elétrica. Uma vez todas as etapas anteriores concluídas é possível prosseguir para o layout, ou projeto físico, atuando na sintetização da descrição geométrica final das máscaras, levando em conta a descrição estrutural do circuito. Desta maneira, o circuito é encaminhado para estágio de fabricação, feito por meio de processos físicos e químicos com base no layout completo, e por fim, na etapa de empacotamento e teste, realiza-se o empacotamento físico para o substrato de silício e o teste do circuito fabricado.

3 Linguagens de Descrição de *Hardware*

Linguagens de descrição de *hardware* (HDLs) são utilizadas com finalidade de projetar *hardwares* através da descrição do comportamento dos mesmos, diferentemente das linguagens de programação tradicionais, que descrevem algoritmos, sequência de operações, de forma serial. Em *hardware*, há o envolvimento de diversos componentes individuais atuando simultaneamente, sendo necessária a capacidade de descrevê-los um a um, bem como suas interligações.

Segundo [Kundert e Chang \(2005\)](#), HDLs tem duas aplicações principais: síntese e simulação. A síntese é o processo de implementação do *hardware*, onde o HDL é usado para descrevê-lo em um nível abstrato usando modelos de componentes que ainda não têm implementação física, criando uma nova descrição. A simulação, por sua vez, é o ato de aplicar estímulos ao modelo executável descrito em HDL, a fim de prever suas reações. Deste modo, pode-se entender como o sistema se comporta antes da implementação, sendo capaz de diminuir custos.

Os HDLs podem descrever sistemas que trabalham com sinais digitais, analógicos ou mistos. Atualmente, existem duas HDLs disponíveis para descrever *hardwares* de sinais mistos: Verilog-AMS e VHDL-AMS. Estas são extensões para os HDLs digitais Verilog e VHDL, com intuito de apoiar a modelagem de sistemas analógicos e mistos. Neste trabalho será utilizada a HDL Verilog-AMS. As seções a seguir são dedicadas à melhor exposição desta linguagem.

3.1 Verilog-AMS

O Verilog-AMS (*Analog and Mixed Signal*) é uma linguagem de descrição de *hardware* oriunda da HDL Verilog, *VERifying LOGic*, onde o AMS destaca a união das linguagens: Verilog-A e Verilog-HDL. Portanto, a linguagem Verilog-AMS permite a descrição de componentes de sinais mistos, podendo especificar sistemas em vários níveis de abstração.



Figura 7 – Divisões do HDL Verilog [[Melnik \(2006\)](#)].

Em um outro ponto de vista, é possível ver a linguagem Verilog-AMS como uma extensão do SPICE, elevando o nível de abstração disponível para os projetistas analógicos [Melnik (2006)], conforme a Fig. (8).

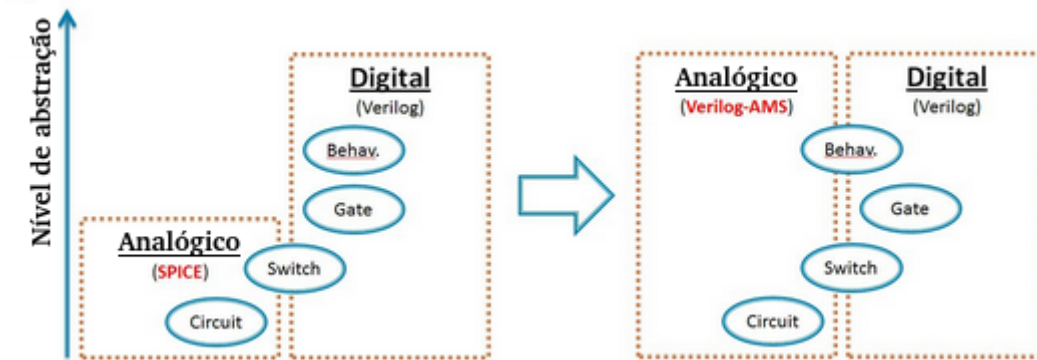


Figura 8 – Visão de uso do Verilog-AMS [Melnik (2006)].

3.1.1 Características

A principal característica da linguagem Verilog-AMS é a capacidade de simplificar projetos considerados complexos, possibilitando a decomposição hierárquica destes, assim como a criação de modelos a partir da descrição de componentes, blocos funcionais e dispositivos, que posteriormente podem ser adicionados aos simuladores. Por Kundert e Chang (2005), outras características interessantes são:

- Criação de *test benches*, ambiente virtual utilizado para verificar o funcionamento ideal do modelo desenvolvido, onde é possível operar as funções do circuito através de uma variedade de comportamentos;
- Aceleração da simulação, isto ocorre através da substituição de partes não críticas por modelos comportamentais, não utilizando modelos a nível de transistores;
- Verificação de sistemas de sinais mistos, permite que ambos os circuitos, digitais e analógicos, sejam descritos mais adequadamente. Ou seja, em circuitos digitais, utiliza-se o Verilog-HDL e em circuitos analógicos, transistores ou Verilog-A, desta forma, as representações podem ser facilmente combinadas. Portanto, os projetistas analógicos e digitais podem operar com a linguagem de preferência, e ainda assim trabalharem juntos;
- Suporte à metodologia de projeto *top-down*, isto é, reduzindo o custo e o tempo necessário para processamento posterior. O Verilog-AMS permite uma forma mais rica de comunicação, onde os modelos podem ser trocados entre os projetistas.

Os códigos em Verilog-AMS seguem um padrão definido, de forma que no corpo principal do programa se encontra as definições de bibliotecas, módulos, declaração dos

pinos, disciplinas, parâmetros, variáveis locais, descrição do comportamento do sistema e as saídas, consecutivamente. Os elementos principais da linguagem estão a seguir.

- *Nature*: é um conjunto de atributos que são compartilhados por uma classe de sinais. Incluem as unidades (*units*), nome usado ao acessar o sinal (*access*), absoluta tolerância (*abstol*) e naturezas relacionados (*ddt nature*, *idt nature*). Exs: *voltage*, *current*, *temperature*, *angle*, *force*, *position*, *acceleration*, *impulse*, *velocity*, etc;
- *Discipline*: é um conjunto de tipos de sinais físicos correlacionados. Pode incluir a especificação de um domínio, contínuo ou discreto, e os atributos definidos nas naturezas de potencial e *flow*. Exs: *logic*, *electrical*, *magnetic*, *thermal*, etc;
- *Include*: é uma diretiva de compilação responsável por acessar arquivos, como por exemplo as bibliotecas, onde seu argumento de entrada é o nome do arquivo de interesse. Outras diretivas de compilação são: *'default discipline* , *'else* , *'resetall* , *'endif* , *'timescale*, *'define* , *'ifdef* , *'default transition* e *'undef*. O caractere *'* introduz as diretivas;
- *Module*: é um bloco de instruções que inicia com a diretiva *module* e finaliza com *endmodule*, o nome do mesmo e as portas envolvidas são apresentadas logo após o início do módulo;
- *Ports*: são pontos onde conexões podem ser feitas com os componentes. São classificados pela direção (*input*, *output* e *inout*) e tipo (ex: *electrical*);
- *Integer/Real/Wreal*: são "tipos de dados abstratos". No caso do *wreal* (*real-wire*) é um tipo de dado real discreto no tempo que pode ser usado para representar tensão e corrente. É muito útil para verificação a nível de sistema, onde blocos analógicos podem ser simulados digitalmente, otimizando a velocidade de simulação;
- *Parameter*: é uma variável, *real* ou *integer*, que a princípio recebe um valor constante ou um intervalo, sendo este valor fixo durante a simulação, podendo ser variado apenas no *testbench*;
- *Analog/Initial/Always*: a diretiva *analog* referencia ao início de um procedimento analógico, contínuo no tempo. Enquanto *always* ou *initial*, introduz um processo digital, discreto no tempo, repetidamente ou não;

A documentação completa sobre a linguagem Verilog-AMS pode ser encontrada em [Kundert e Chang \(2005\)](#). No apêndice (C) é documentado um exemplo de modelagem em Verilog-AMS de um conversor AD de 16 bits utilizando ferramentas Cadence.

4 *Phase Locked Loop*

Phase Locked Loop (PLL), ou também chamado de Malha de Travamento de Fase, é uma parte fundamental das tecnologias referentes ao rádio, transmissões sem fio e telecomunicações. É usado para as mais diversas finalidades, como a regeneração de sinais, demodulação FM e sincronismo em transmissões digitais.

O PLL é um caso particular de um sistema negativamente realimentado, podendo ser implementado analógica ou digitalmente, de forma a permitir a geração estável e com baixo ruído de sinais RF sintonizáveis. Isto ocorre através de multiplicações em frequência a partir de uma referência comum, um exemplo desse referencial de frequência é o TCXO (*Temperature Compensated Crystal Oscillator*), possibilitando ao circuito sintetizador a geração de uma vasta gama de sinais.

Segundo [Correa e Paolo \(2011\)](#), a diferença básica entre PLLs analógicos e digitais são seus componentes. PLLs digitais ou DPLLs, trabalham em tempo discreto e possuem PFD e utilizam filtros digitais. Outro termo existente para DPLLs é ADPLL, usado para aqueles que são inteiramente digitais, geralmente implementados em *hardware*. Já os PLLs analógicos, compostos geralmente por detector de fase, filtro de malha e VCO geram sinais senoidais. Entretanto, PLLs podem conter blocos mistos, analógicos e digitais, desta forma, a principal razão para que eles possam ser classificados como analógicos ou digitais é que a taxa de amostragem do sistema PLL em relação a sua largura de banda.

O projeto de PLL visa manter um compromisso entre economia de espaço, potência e custo, em conjunto com uma pureza espectral. Desta forma, os PLLs assumem suas funções em circuitos altamente integrados, trabalhando com sinais digitais e mistos, e operando em baixo consumo [[Barrett \(1999\)](#)].

4.1 Funcionamento do PLL

O princípio básico de funcionamento do PLL consiste na correção contínua da diferença de fase e/ou frequência existente entre os dois sinais da entrada do *loop*. Isto ocorre através da interação entre 4 blocos básicos (Fig. 9): detector de fase/frequência (PD/PFD), filtro de malha, oscilador controlado por tensão (VCO) e divisor de frequência.

O PD fornece uma tensão de saída cuja componente DC é proporcional a diferença de fase/frequência entre o sinal de entrada e sinal do VCO. Este sinal gerado pelo PD é encaminhado ao filtro, onde é extraída a sua componente contínua, tensão utilizada como sinal de controle do oscilador. O VCO, por sua vez, é responsável por gerar um sinal cuja frequência é dependente da tensão de controle, de forma que este sinal será posteriormente

dividido pelo bloco divisor de frequência e realimentado no *loop*, como entrada do PD. Observa-se que a frequência do sinal de saída do VCO é uma multiplicação da frequência do sinal de referência.

Numa situação onde não existe um sinal de entrada, a frequência do sinal de saída é determinada apenas pelo VCO, a partir de suas características, permanecendo num valor central.

Quando uma entrada (Ref) é aplicada o PD realiza a comparação entre a entrada e o sinal gerado pelo VCO. Sendo diferentes, gera-se um sinal proporcional a essa diferença que passará pelo filtro e servirá como tensão de controle do VCO, ajustando a frequência do sinal de saída deste bloco, aproximando-a da frequência do sinal de entrada. A partir do momento que estas se igualam, o VCO “trava”, ou seja, diz-se que capturou o sinal de entrada. Neste momento, a saída do PLL é um sinal multiplicado em frequência por um fator N . Uma alteração na frequência do sinal de entrada acarretará num novo sinal diferença, gerado na saída do PD, causando uma mudança de tensão na saída do filtro que levará o VCO a se adaptar a essa nova frequência [Bistue, Quemada e Adin (2009)].

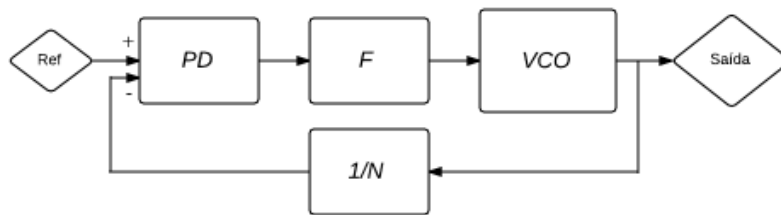


Figura 9 – PLL básico [Bistue, Quemada e Adin (2009)].

4.2 Componentes do PLL

A arquitetura de um PLL mais completo consiste na presença de pelo menos 5 blocos: detector de fase/frequência (PD/PFD), *charge pump*, filtro de malha, oscilador controlado por tensão (VCO) e o divisor. Em abordagens CMOS, é comum que o *charge pump* seja integrado com o PFD.

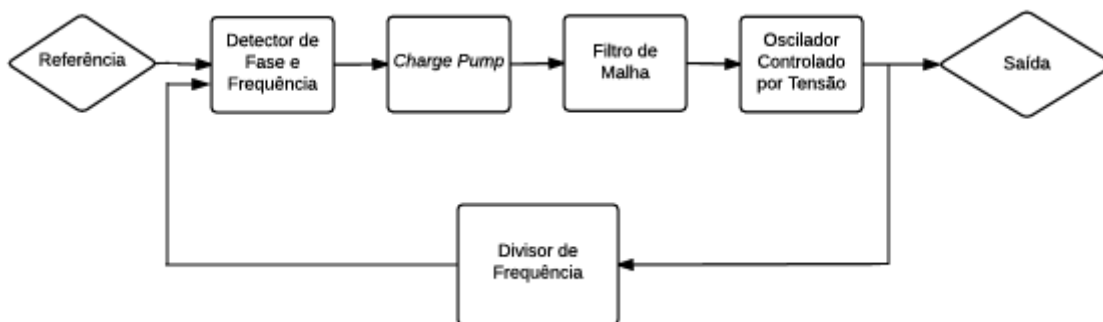


Figura 10 – Diagrama de blocos do PLL [Dabhi e Nagpara (2014)].

4.2.1 PD/PFD

O bloco do detector de fase/frequência (PD/PFD), é responsável por gerar um sinal de erro, fruto da comparação entre o sinal de entrada (Ref) e o sinal proveniente do VCO, após ser dividido no bloco divisor. Vale ressaltar que os circuitos PDs realizam apenas a comparação de fase, enquanto os PFDs, fase e frequência, tendo este segundo vantagens sobre o primeiro, mais informações sobre este fato podem ser encontradas em [Razavi e Behzad \(1998\)](#). A Fig.(11) exemplifica o funcionamento básico de um PFD/CP.

4.2.2 Charge Pump (CP)

Charge Pump é o bloco que geralmente acompanha o PFD e possui a função de converter o estado lógico produzido no PFD num sinal analógico adequado para controle do VCO, através da conversão da saída deste numa fonte de pulsos de corrente. A Fig.(11) indica a corrente de saída do CP em conjunto com o PFD e o filtro de malha.

4.2.3 Filtro de Malha

O filtro de malha é responsável por filtrar a saída do CP e converter os pulsos de corrente num valor contínuo de tensão, sendo esta tensão usada para controlar o VCO. Ou seja, o filtro de malha funciona como uma rede de transimpedância, convertendo corrente em tensão enquanto filtra. Este bloco também está relacionado ao aspecto de estabilidade da realimentação, atenuação de espúrios indesejados e na determinação da largura de banda do *loop*, aspecto influente no ruído total do PLL.

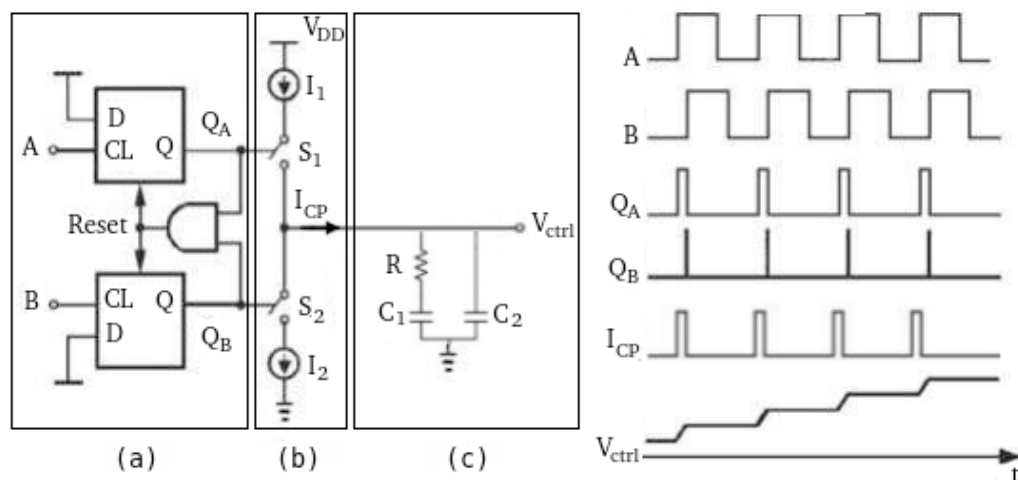


Figura 11 – PFD/CP e Filtro de Malha: (a) PFD; (b) *Charge Pump*; (c) Filtro de Malha [[Henzler \(2011\)](#)].

4.2.4 VCO

VCO ou oscilador controlado por tensão, é o bloco responsável por gerar um sinal de saída cuja frequência é proporcional a uma tensão de controle proveniente do *loop* de realimentação do PLL, a fim de ajustar a frequência de saída desse bloco com a frequência do sinal de referência (Ref) (Fig. 12). Ou seja, a sua saída é utilizada para sintonização, sendo uma das entradas do PFD.

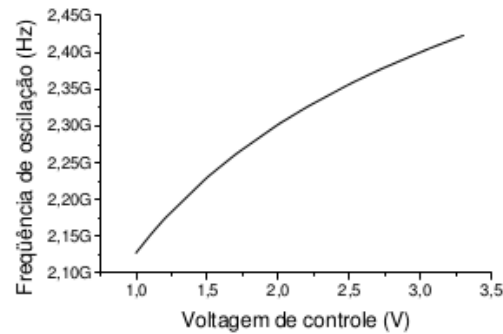


Figura 12 – Resposta do VCO a partir da tensão de controle [Argüello (2004)].

4.2.5 Divisor

O divisor realiza o *loop* de realimentação do PLL. A sua função básica é reduzir a frequência do VCO dentro de uma faixa de valores que podem ser comparadas com o sinal de referência, ou seja, é responsável pela seleção do canal. A simulação abaixo corresponde ao funcionamento do divisor encontrado na arquitetura N-inteiro, sendo composto pelo *swallower counter* e o *dual-modulus prescaler*, capaz de gerar múltiplos inteiros da frequência do sinal de referência.

4.3 Parâmetros do PLL

Visando o correto funcionamento do PLL, alguns parâmetros devem ser levados em consideração para que minimize as ações de eventuais problemas de projeto. Segundo Barrett (1999), alguns destes parâmetros são:

- Faixa de Frequência: frequência de operação do PLL;
- Resolução de Frequência: menor incremento de frequência possível;
- Ruído de Fase: indicador da qualidade do sinal;
- Nível de Sinal Espúrio: medida da interferência discreta, determinística no espectro do sinal;
- Largura de Banda de *Loop*: medida da velocidade dinâmica do loop;

- Tempo de Travamento ou *settling time*: tempo necessário para estabelecer uma nova frequência, ou seja, sintonizar um novo canal.

4.4 Tipos de Arquitetura

Dependendo do valor N utilizado no bloco divisor, o PLL pode ser classificado como inteiro ou fracionário. As seções a seguir introduzem o conceito e aspectos principais de cada uma destas arquiteturas.

4.4.1 N-Inteiro

Este tipo de sintetizador de frequência gera como saída a frequência F_{OUT} calculada de acordo com a Eq.(4.1), onde N varia entre N_L e N_H .

$$F_{out} = N \cdot F_{ref} \quad (4.1)$$

Observa-se que a frequência de referência (F_{REF}) deve coincidir com o espaçamento entre os canais, a fim de permitir uma seleção consistente. Desta forma, quando N assume o valor N_L o canal inferior é selecionado. Por meio da variação de N , o restante dos canais são sintonizados, sucessivamente, até que o canal superior é alcançado, onde N atinge o valor N_H . Um exemplo deste tipo de divisor é o *pulse-swallow* (Fig. 13). Mais informações podem ser encontradas em [Razavi e Behzad (1998)].

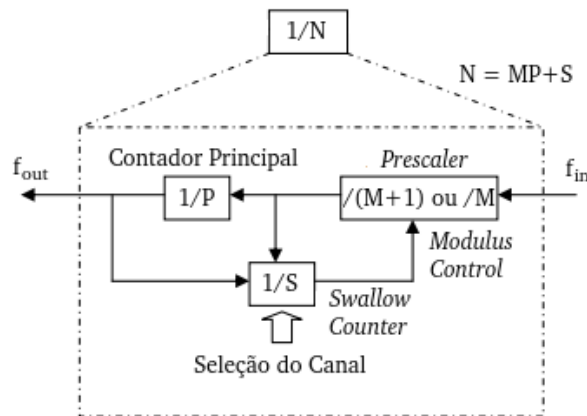


Figura 13 – Divisor de frequência *Pulse-Swallow* [Argüello (2004)].

As vantagens desta arquitetura são: simplicidade, baixo consumo de energia, baixo custo e economia de espaço. Entretanto, as desvantagens incluem aparição de espúrios no sinal de saída e limitação da largura de banda do *loop* [Barrett (1999)].

Os espúrios são observados a uma distância F_{REF} da frequência da portadora, podendo ser atenuados através de um filtro de malha adequado. Em relação a limitação da largura de banda do *loop*, esta ocorre devido a necessidade da F_{REF} coincidir com o

espaçamento entre os canais, levando a um compromisso entre o *settling time*, emissão de espúrios e ruído de fase.

Por [Bistue, Quemada e Adin \(2009\)](#), ao passo que espúrios aparecem a uma distância F_{REF} da portadora, uma largura de banda muito superior a esta frequência pode atenuar estes sinais indesejados, porém deixa o sistema mais lento, e vice-versa. O mesmo ocorre para o ruído de fase, o compromisso deste com a velocidade de sintonização é necessário para determinação da largura de banda.

4.4.2 N-Fracionário

Esta arquitetura é muito similar a N-inteiro, porém existe a adição do bloco acumulador. O acumulador é uma máquina de estados que muda o fator de divisão durante o estado *locked*, variando-o dinamicamente entre N e $N+1$, possibilitando a geração de um número fracionário. A geração de um número fracionário resolve o problema de limitação da largura de banda do *loop*, presente no N-inteiro.

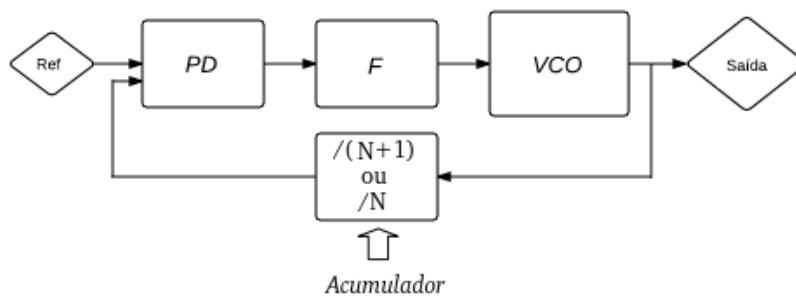


Figura 14 – Arquitetura N-fracionário [[Argüello \(2004\)](#)].

Como vantagens, esta arquitetura apresenta diminuição da limitação da largura de banda do *loop*, resposta transitória rápida em conjunto com supressão do ruído de fase do VCO e redução do efeito do ruído de fase do PD. Por outro lado, suas desvantagens são: aparição de espúrios em frequências fracionárias ao sinal de referência, aumento no ruído de fase e complexidade do circuito.

As vantagens apresentadas acima são consequências da diminuição da necessidade de que F_{REF} coincida com a separação dos canais. No entanto, esta arquitetura apresenta o mesmo problema da outra quanto a aparição de espúrios, porém, mais crítico, pois os níveis destes sinais são superiores aos da N-inteiro, sendo necessária a adição de circuitos para compensar este fato. Os circuitos adicionais elevam o nível de ruído de fase e aumentam a complexidade do circuito.

4.5 Ruído de Fase

Este fenômeno pode ser definido como um desvio aleatório da frequência da portadora que se espalha em torno da frequência central. Em aplicações RF, a portadora ideal é geralmente caracterizada por um impulso, onde na prática ocorre um espalhamento no domínio das frequências, afetando as frequências laterais à central.

Considerando um PLL como um sistema de realimentação negativa no domínio s (Fig. 9) é interessante a representação do mesmo como uma função de transferência $G(s)$, onde $H(s)=1/N$ é o ganho de realimentação. Para malha aberta, tem-se o seguinte ganho:

$$G(s) = \frac{K_{\Phi} \cdot K_{VCO} \cdot Z(s)}{s} \quad (4.2)$$

Onde K_{Φ} é o valor de corrente do *charge pump* em amperes, localizado na saída do PFD, K_{VCO} é o ganho do VCO em Hz/V, N é o módulo do divisor e $Z(s)$ é a função de transferência do filtro. Portanto, o ganho de malha fechada é:

$$F(s) = \frac{N \cdot K_{\Phi} \cdot K_{VCO} \cdot Z(s)}{N \cdot s + K_{\Phi} \cdot K_{VCO} \cdot Z(s)} \quad (4.3)$$

As funções de transferência do ruído de cada bloco podem ser derivadas encontrando-se a relação entre a resposta na saída e a contribuição de entrada correspondente. A Tabela (2) fornece as contribuições de ruído de cada bloco, maiores informações estão disponíveis em [Banerjee (2006)].

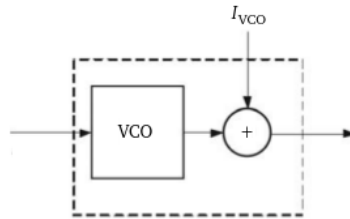


Figura 15 – Diagrama usado para computar as contribuições de ruído de cada bloco [Bistue, Quemada e Adin (2009)].

O ruído de fase total na saída do PLL é obtido multiplicando cada fonte pela sua função de transferência e adicionando todos estes produtos resultantes. A fim de simplificar a expressão, consideram-se fatores comuns entre as fontes de ruído como $A(s)$ e $B(s)$, onde $H=1/N$ e $G(s)$ foi definido anteriormente na Eq.(4.2).

$$A(s) = \frac{G(s)}{1 + H \cdot G(s)} \quad (4.4)$$

$$B(s) = \frac{1}{1 + H \cdot G(s)} \quad (4.5)$$

Tabela 2 – Funções de transferência das fontes de ruído do PLL [Banerjee (2006)].

Fonte do ruído	Função de transferência
Referência (I_{XTAL})	$\frac{G(s)}{1+H \cdot G(s)}$
Divisor N (I_N)	$\frac{G(s)}{1+H \cdot G(s)}$
Detector de fase/frequência (I_{PFD})	$\frac{1}{K_\Phi} \cdot \frac{G(s)}{1+H \cdot G(s)}$
Filtro de Malha (I_{LF})	$\frac{\sqrt[3]{2} \cdot K_{VCO}}{2 \cdot f} \cdot \frac{G(s)}{1+H \cdot G(s)}$
VCO (I_{VCO})	$F(s) = \frac{1}{1+H \cdot G(s)}$

Uma expressão para o ruído de fase do PLL, em dBc/Hz, é apresentada a seguir, lembrando que $s = j\omega$ [Bistue, Quemada e Adin (2009)].

$$L(\omega) = 10 \cdot \log \left[I_{XTAL}(\omega) \cdot |A(s)|^2 + I_{VCO}(\omega) \cdot |B(s)|^2 + I_{PFD}(\omega) \cdot F_{REF} \cdot |A(s)|^2 \right. \\ \left. + 2 \cdot k \cdot T \cdot \frac{K_{VCO}^2}{f^2} \cdot |T(s) \cdot B(s)|^2 \right] \quad (4.6)$$

Onde:

k : constante de Boltzmann;

T : temperatura absoluta;

ω : frequência de *offset* com respeito a frequência de entrada do PLL;

$T(s)$: função de transferência da tensão ruído das resistências do filtro de malha;

I_{XTAL} : ruído de fase da referência;

I_{VCO} : ruído de fase do VCO;

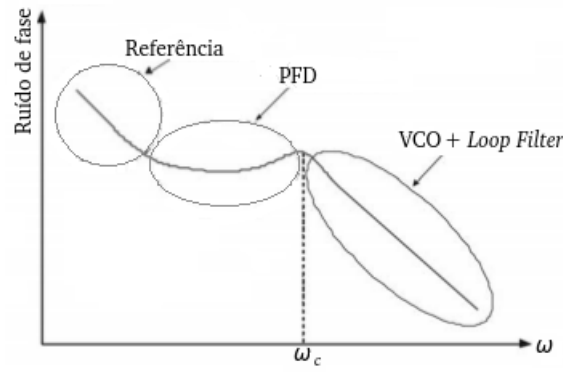


Figura 16 – Ruído de fase típico do PLL [Bistue, Quemada e Adin (2009)].

I_{PFD} : ruído de fase do PFD.

Pode ser visto que para baixas frequências a fonte de ruído dominante é oriunda da frequência de referência. Para frequências um pouco mais altas, mas ainda menores que a largura de banda loop (ω_c), o ruído do PFD predomina. Por fim, em frequências mais altas que ω_c , as fontes de ruído são o VCO e o filtro de malha.

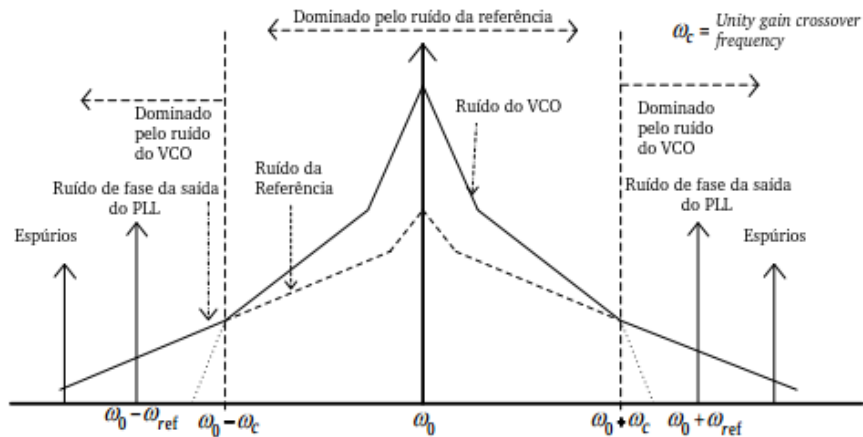


Figura 17 – Ruído de fase típico da saída do PLL [Manthana (2011)].

4.6 Emissão de Espúrios

Outro efeito não ideal que impacta o funcionamento do PLL é a emissão de sinais espúrios. Considera-se espúrio qualquer sinal indesejado, como por exemplo, harmônicos ou sinais externos ao transmissor. Existem duas causas fundamentais para origem de espúrios no PLL: correntes de fuga e desencontros no *charge pump*.

Em baixas frequências, cerca de KHz, os efeitos causados por correntes de fuga são as principais causas de emissão de espúrios. Tais emissões são naturais das características intrínsecas das junções PN, estrutura fundamental de semicondutores, como o transistor.

Dessa forma, estas correntes parasitas causam modulações errôneas no VCO, gerando sinais indesejados.

O *charge pump*, em *lock state*, permanece inativo a maior parte do tempo. Os períodos de atividade são muito curtos, gerando impulsos de corrente, o que não muda a tensão de controle do VCO, mas causa *jitter*, desvio da periodicidade, na sintonização. São estes pequenos impulsos responsáveis por gerar espúrios. Algumas das causas destes impulsos são: o descasamento entre as correntes de carga e descarga do *charge pump*, a diferença entre os tempos de ativação dos transistores PMOS e NMOS e o circuito de eliminação da *dead zone* presente no PFD. Mais informações em [Banerjee (2006)].

4.7 Proposta de Topologia de PLL para ZigBee

Para este projeto, será desenvolvido um PLL com saídas analógicas diferenciais. Em relação a arquitetura do sintetizador, após efetuar a pesquisa bibliográfica sobre topologias de PLL utilizadas para aplicações em GHz, mais especificamente para o ZigBee, a mais recorrente foi a N-inteiro, apresentada anteriormente. Portanto, o projeto terá como objetivo a modelagem em alto nível utilizando linguagem Verilog-AMS de um PLL diferencial com arquitetura N-inteiro.

5 Topologias de VCO e Proposta de Circuito

O circuito oscilador pode ser visto como um sistema de dois terminais realimentado positivamente ou como um sistema composto por um bloco de circuito ativo e um ressonador.

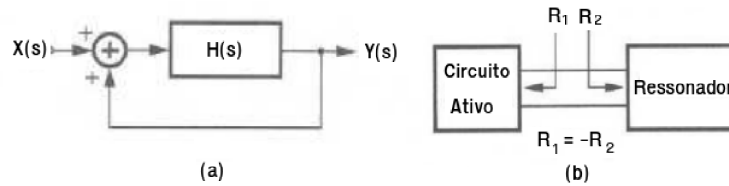


Figura 18 – Modelos de osciladores: (a) Sistema linear com realimentação positiva; (b) Modelo de resistência negativa [Razavi e Behzad (1998)].

Segundo o critério de Barkhausen, o sistema permanecerá oscilando com frequência $s = j\omega_0$ quando satisfeitas as seguintes condições [Farfán (2003)]:

$$|H(j\omega_0)| \geq 1 \quad E \quad \angle H(j\omega_0) = 180^\circ \quad (5.1)$$

Sendo este critério necessário mas não suficiente, na prática, escolhe-se $|H(j\omega_0)|$ algumas vezes maior que o necessário. Outro aspecto é relacionado ao LGR, onde para ocorrer instabilidade, necessita-se da presença de apenas um pólo no semi-plano direito, entretanto, em condições reais, para que um sistema oscile deve haver um par de pólos complexos conjugados no semi-plano direito [Madureira (2008)].

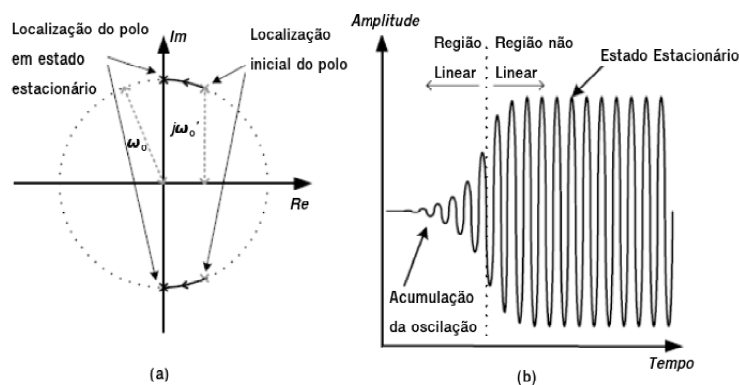


Figura 19 – (a) LGR do oscilador; (b) Forma de onda de saída [Berny et al. (2006)].

Um oscilador controlado por tensão (VCO) é um circuito que fornece um sinal de saída variável, cuja frequência pode ser ajustada, ao longo de uma faixa, através do controle dado por um sinal de tensão DC.

Por [Henzler \(2011\)](#), o VCO produz uma frequência de saída que pode ser aumentada ou diminuída de acordo com uma tensão de entrada, segundo a seguinte relação:

$$f = f_{fr} + K_{KVO} \cdot V_{in} \quad (5.2)$$

Onde:

f_{fr} : frequência de funcionamento;

K_{VCO} : fator de ganho do VCO;

V_{in} : tensão de entrada.

5.1 Parâmetros do VCO

Alguns parâmetros são estipulados para verificar o funcionamento do VCO, segundo [Razavi e Behzad \(1998\)](#) e [Kinet \(1999\)](#), alguns destes parâmetros são:

- Frequência Central: frequência de saída f_0 do VCO quando a tensão de controle está em seu valor central;
- Faixa de Sintonia: intervalo de frequências de saída no qual o VCO oscila ao longo da extensão da tensão de controle;
- Sensibilidade de Sintonia: é a mudança na frequência de saída por unidade de variação na tensão de controle, normalmente expressa em [Hz/V];
- *Load Pulling*: quantifica a sensibilidade da frequência de saída para mudanças na sua carga;
- *Supply Pulling*: quantifica a sensibilidade da frequência de saída a alterações na tensão de alimentação, expressa em [Hz/V];
- Supressão Harmônica: especifica o quanto os harmônicos do sinal de saída são menores comparados ao componente fundamental, expresso em [dBc];
- Pureza Espectral: relação do espectro com ruídos, pode ser especificada no domínio do tempo em termos de *jitter* ou no domínio da frequência em termos de ruído de fase.

5.2 Tipos de VCO

Idealmente, o VCO deve gerar um sinal sem ruído de fase, ser sintonizado numa faixa fixa de frequência, ser insensível à carga de saída, a variações de temperatura, de processos de fabricação e tensão de alimentação. No entanto, este comportamento está distante do real [Farfán (2003)].

Alguns dos tipos de osciladores existentes estão apresentados na Fig.(20), estes são: oscilador a cristal, de relaxação, em anel de inversores e LC. Este último é a melhor opção para este projeto, desta forma será melhor abordado a seguir, as considerações da motivação de seu uso serão abordadas nas sessões subsequentes.

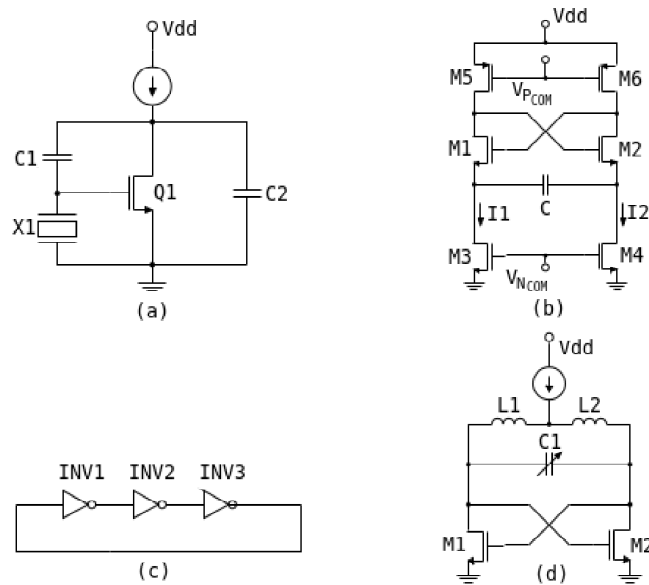


Figura 20 – Topologias de osciladores: (a) oscilador a cristal; (b) oscilador de relaxação; (c) oscilador em anel com inversores; (d) oscilador LC [Farfán (2003)].

5.2.1 Cristal

Osciladores à cristal utilizam a resposta em frequência de um cristal pizeoelétrico para selecionar a frequência de operação do circuito. Sua maior vantagem é a pureza espectral, no entanto, como desvantagem, tem-se uma dependência da frequência com o cristal associado. Este oscilador possui alcance relativamente baixo, algumas dezenas de MHz, sendo inadequados para aplicações em RF, entretanto, devido a sua boa performance em relação à ruídos, estes osciladores são comumente usados como frequência de referência de PLLs [Madureira (2008)].

5.2.2 Relaxação

O oscilador de relaxação possui funcionamento relacionado com o carregamento alternado do capacitor C pelas correntes I_1 e I_2 . A frequência de oscilação é modificada

variando-se a corrente de carga. A principal desvantagem desta topologia é o alto consumo, sendo este alto consumo necessário para diminuição do ruído de fase do mesmo [Farfán (2003)].

5.2.3 Em anel

O oscilador em anel com inversores é considerado o mais simples e integrável. Modifica-se a frequência através do controle dos atrasos dos inversores, administrando o valor da corrente que comanda a polarização dos transistores. Contudo, apesar da simplicidade e de ser facilmente integrável, esta classe de osciladores possui altos níveis de ruído de fase, devido a constante comutação dos inversores, tornando-se uma má opção para aplicações em RF [Farfán (2003)].

5.2.4 Tanque LC

Osciladores LC podem ser vistos a partir da abordagem de resistência negativa (Fig. 18(b)), ou seja, é um circuito de única saída dividido em três elementos: circuito ativo, circuito ressonante e realimentação positiva.

Este circuito gera um sinal de saída periódico a partir da ressonância. O tanque LC, idealmente, determina a sua frequência de oscilação a partir da Eq.(5.3). Todavia, apresenta-se perdas, portanto o circuito amplificador, através da realimentação, é útil para injetar ao tanque LC a energia necessária para compensar tais perdas e permitir que a oscilação seja mantida.

$$f_{osc} = \frac{1}{2\pi\sqrt{LC}} \quad (5.3)$$

Como o capacitor e o indutor são elementos não ideais, efeitos parasitas são observados produzindo perdas e atenuações no funcionamento ideal do circuito tanque LC. Segundo Bistue, Quemada e Adin (2009), pode-se representar estas perdas como uma condutância G_P em paralelo ao tanque, deteriorando a oscilação de sua saída, como representado na Fig.(21).

Sistemas integrados com aplicações em alta frequências sofrem bastante devido aos efeitos parasitários apresentados por estes componentes, uma vez que para frequências acima de 1 GHz tais efeitos podem ser tão representativos quanto os valores dos componentes do tanque, logo é interessante atentar aos possíveis modelos de uso, ao fator de qualidade e as configurações.

No caso dos indutores integrados, pode-se utilizar: modelo Π , no qual apresenta relativa simplicidade, indutores convencionais em configuração espelho, muito usados para

topologias NMOS diferenciais, e indutores balanceados, sendo estes geometricamente simétricos e usados para arquiteturas CMOS.

Para o componente capacitivo do tanque LC, a implementação integrada pode ser feita com transistores configurados em *diode-connected*, interligando a porta com o dreno, ou varactores, estes definidos como capacitores variáveis ajustados a partir da tensão aplicada, sendo uma solução simples e compatível com os processos de integração da tecnologia CMOS.

Os diferentes tipos de circuitos tanque LC se diferem basicamente pela forma de geração da resistência negativa usada para compensar as perdas no tanque. Esta resistência negativa, representada como condutância negativa G_M , deve ser necessariamente maior, em módulo, que a condutância positiva G_P , para garantir a manutenção da oscilação do tanque.

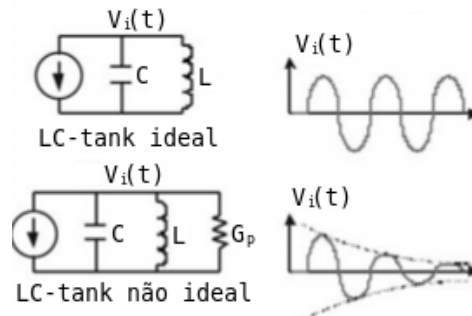


Figura 21 – Resposta transiente do tanque LC ideal e não ideal [Bistue, Quemada e Adin (2009)].

5.2.4.1 Tipos de VCO tanque LC

Como neste projeto será utilizada a tecnologia CMOS, as opções de topologias tanque LC são baseadas no tipo de transistor utilizado, podendo ser NMOS, PMOS e a combinação dos dois, CMOS. No que diz respeito a característica do sinal de saída, podem ser classificados como *single-ended* ou diferencial. As principais vantagens do diferencial em relação ao *single-ended* são relacionadas a alta rejeição do ruído de modo comum, melhor isolamento, atenuação de harmônicos, diminuição da dependência do circuito à variáveis externas e melhoria no ruído de fase. Porém, esta abordagem necessita do dobro de componentes, impactando na área de *chip*, e consome aproximadamente o dobro da potência.

NMOS. Nesta configuração, conforme a Fig.(22), tem-se o circuito ativo formado por um par cruzado de transistores NMOS. As vantagens são simplicidade, devido ao número reduzido de transistores, e conseqüentemente o baixo ruído que contribui com alto nível de linearidade. Por outro lado, este circuito possui um consumo relativamente alto em

consequência da quantidade de transistores, sendo necessária uma corrente de polarização alta para produzir a resistência negativa correta, arriscando-se ser muito alta para dispositivos de baixo consumo.

Existem 3 principais topologias de tanque LC com NMOS, basicamente elas se diferem na forma de geração da corrente de polarização, por fonte de corrente ou por um resistor, e na presença de um capacitor em paralelo com a fonte de corrente que auxilia no aterramento da fonte do NMOS.

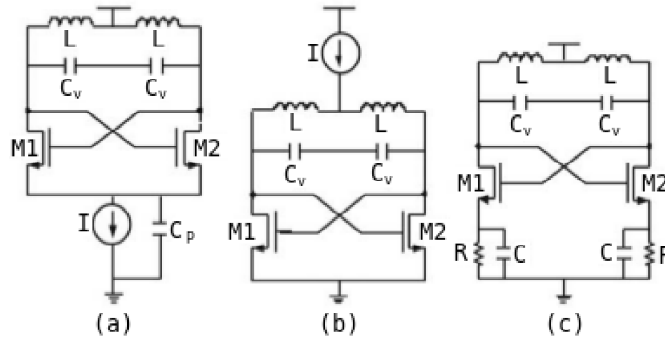


Figura 22 – (a-c) Topologias tanque LC NMOS diferenciais [Bistue, Quemada e Adin (2009)].

PMOS. As configurações dos osciladores tanque LC constituídos por transistores PMOS são similares aos anteriores, a diferença está no uso de transistores PMOS no lugar dos NMOS e nos devidos rearranjos. A principal desvantagem desta topologia, encontra-se na performance destes transistores comparados aos NMOS, geralmente pior, onde necessita de maior área para alcançar a mesma resistência negativa com o mesmo consumo de potência, devido a menor mobilidade dos portadores de carga. Portanto, estas configurações não são muito usuais.

CMOS. Com a utilização de transistores NMOS e PMOS, os tanque LC CMOS utilizam 2 transistores de cada tipo para obter a regeneração do sinal com a mesma amplificação gerada com as abordagens anteriores. Isto ocorre com uma menor demanda de corrente de polarização, sendo esta a sua principal vantagem, resolvendo o problema de consumo. Entre suas desvantagens, devido ao acréscimo de 2 transistores, é possível citar: aumento de complexidade, necessidade de maior tensão de alimentação, maior ruído para mesma corrente de polarização, maior consumo de área e menor faixa de variação da frequência do sinal. Este último se dá pela adição de capacitância parasita ao circuito e pela diminuição da faixa de valores da tensão de controle do VCO.

As topologias da Fig.(23) são as soluções de tanque LC CMOS, as mesmas são similares as do NMOS, ou seja, difererem-se através da forma com que a corrente de polarização é gerada e/ou com a presença ou não do capacitor em paralelo com a fonte de corrente.

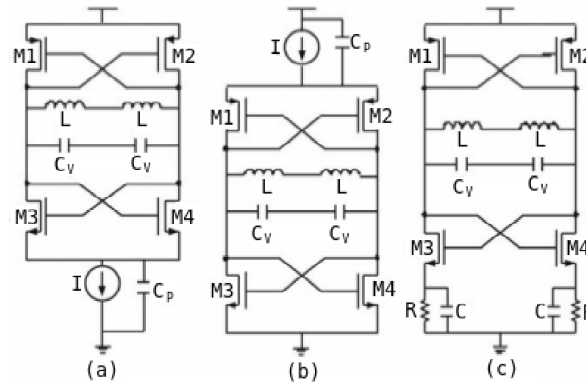


Figura 23 – (a-c) Topologias tanque LC CMOS diferenciais [Bistue, Quemada e Adin (2009)].

5.3 Ruído de Fase no VCO

O espectro de saída do VCO real consiste num tom fundamental, localizado na frequência de oscilação, e componentes indesejados de frequência, responsáveis pela aparição de bandas laterais e harmônicos.

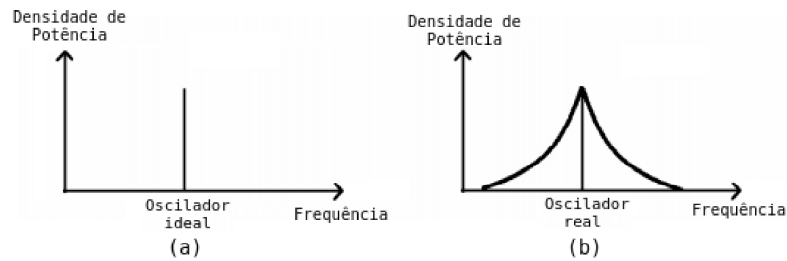


Figura 24 – Espectro de frequência do oscilador: (a) Ideal; (b) Real [Anjos (2012)].

O sinal de saída de um oscilador real pode ser representado pela Eq.(5.4), onde $A(t)$ é a amplitude e $\theta(t)$ a fase. Pelo fato de $A(t)$ e $\theta(t)$ serem funções dependentes do tempo, bandas laterais aparecem no espectro de saída do oscilador em volta da frequência de oscilação ω_0 .

$$V_{out}(t) = A(t) \cdot \sin[\omega_0 \cdot t + \theta(t)] \quad (5.4)$$

Os fenômenos que aparecem no espectro real da saída do oscilador geram efeitos negativos no funcionamento de transceptores, onde a presença de bandas laterais em torno da frequência de oscilação expõe o poder de ruído nas proximidades da portadora. As imagens da Fig.(24) exemplificam bem um problema neste âmbito.

De acordo com a Fig.(25), num receptor é necessário converter um canal de frequência localizada no ω_1 para outra frequência ω_2 , porém existem canais vizinhos a uma distância ω_{ad} , conforme Fig.(25)(a). Após a conversão (Fig. 25(b)), os canais exibem o efeito do ruído de fase, desta forma, seus espectros se espalharam ao longo da frequência

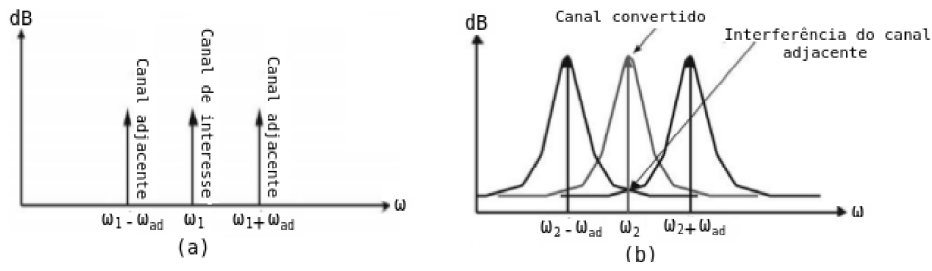


Figura 25 – Espectro de frequência do oscilador com canais adjacentes: (a) Ideal; (b) Real [Bistue, Quemada e Adin (2009)].

central. Portanto, o sinal de interesse recebe sinais indesejados, elevando o nível de ruído do canal.

Existem modelos que desenvolvem análises quantitativas e/ou qualitativas do ruído de fase provocado pelo VCO, explorando os domínios de frequência e temporal. Alguns modelos são: Leeson, Craninckx, Hajimiri e Lee.

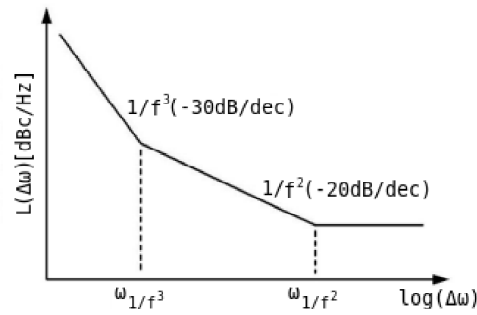


Figura 26 – Representação gráfica do modelo de Leeson [Farfán (2003)].

Para Leeson (1966), analisando no domínio da frequência e assumindo o oscilador como um sistema linear invariante no tempo, uma estimação do ruído de fase (em dBc/Hz) da saída do oscilador pode ser feita com a equação a seguir.

$$L(\Delta\omega) = 10 \cdot \log \left\{ \frac{2FkT}{P_{carrier}} \cdot \left[1 + \left(\frac{\omega_0}{2Q\Delta\omega} \right)^2 \cdot \left(1 + \frac{\omega_1/f^3}{|\Delta\omega|} \right) \right] \right\} \quad (5.5)$$

Onde:

k: constante de Boltzmann;

T: temperatura absoluta;

Q: fator da qualidade tanque LC;

ω_0 : é a frequência de oscilação;

$\Delta\omega$: deslocamento de frequência em relação a portadora;

$P_{carrier}$: potência da portadora;

F: fator empírico que leva em conta o aumento da densidade de ruído na região $(1/\Delta\omega)^2$;

$\omega_{(1/f^3)}$: frequência que limita as regiões $(1/\Delta\omega)^2$ e $(1/\Delta\omega)^3$.

Observa-se neste modelo que o ruído de fase do VCO diminui quando a potência da portadora, o fator de qualidade do tanque e/ou o deslocamento de frequência ($\Delta\omega$), no que diz respeito a portadora, aumentam.

5.4 Proposta de Topologia do VCO

No âmbito do PLL, o oscilador juntamente com o divisor são os blocos que apresentam maiores dificuldades no projeto, o principal motivo é o fato destes blocos serem compostos por componentes analógicos e operam em frequências muito elevadas. Portanto, no contexto da aplicação em RF, a topologia tradicionalmente usada e mais adequada para implementar o oscilador integrado é a tanque LC diferencial.

Dentre as topologias apresentadas, descartando-se os tanque LC PMOS por motivos já apresentados, é possível o uso dos tipos NMOS e CMOS. Portanto, as topologias mais adequadas para implementações integradas na faixa de frequência e aplicação em questão são as apresentadas nas Fig.(22) e (23), similares as da Fig.(27).

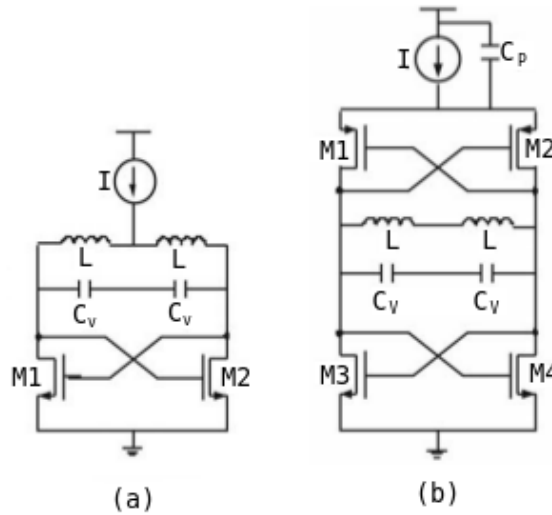


Figura 27 – Topologias de VCO propostas: (a) NMOS; (b) CMOS [Bistue, Quemada e Adin (2009)].

Parte II

Projeto, Implementação e Resultados

6 Projeto do VCO

Para o projeto do VCO, primeiramente é preciso escolher a topologia que atenda as especificações desejadas segundo a aplicação. Na Fig.(27), apresentou-se duas topologias interessantes para a aplicação em questão. Portanto, partindo destas duas topologias e levando em conta diversas referências de trabalhos com PLL para aplicações em ZigBee, chegou-se na definição da topologia de VCO diferencial a ser usada no projeto, apresentada na Fig.(28). As especificações e o procedimento seguido para o projeto elétrico do VCO estão a seguir, para maiores informações, [Hamel \(2005\)](#).

6.1 Especificações para projeto do VCO

Por [Hamel \(2005\)](#), para iniciar o projeto do VCO, algumas especificações devem ser dadas.

1. Máximo consumo de potência

Para o V_{DD} igual a 1.8 V e assumindo I_{BIAS} aproximadamente 2.2 mA, temos:

$$P_{MAX} = V_{DD} \cdot I_{BIAS} = 1.8 \cdot 2.2 \cdot 10^{-3} = 3.96 \text{ mW}. \quad (6.1)$$

2. Estabelecer a excursão do sinal de saída *single ended*

Assumi-se V_{TANQUE} igual a 550 mV.

3. Cálculo do percentual da faixa de sintonização

Sabendo que a faixa de sintonização do VCO é de 2.35 GHz a 2.525 GHz, com frequência central em 2.4375 GHz, temos:

$$\text{Percentual da faixa de sintonização} = \frac{2.525 - 2.35}{2.4375} \cdot 100 = 7.179 \%. \quad (6.2)$$

4. Ganho de malha fechada ($\alpha_{MIN} > 1$)

Os valores usuais de α_{MIN} são 2 ou 3, desta forma, assumi-se α_{MIN} igual a 2.

5. Área do *chip*

Não se tem uma estimativa de área para o VCO, por tal motivo, tenta-se realizar o projeto com o mínimo possível, ou seja, usar os menores valores de indutor, sendo que este é o componente que mais impacta em relação a área.

6.2 Procedimento de projeto do VCO

1. Estipular o valor de I_{bias}

Como dito anteriormente, foi usado I_{bias} igual a 2.2 mA.

2. Determinar o máximo fator de qualidade Q_L do indutor para a dada frequência de operação ω_0

A fim de aproximar os cálculos com os valores práticos, assumi-se Q_L igual 7.8, onde este fator de qualidade é dado pelo próprio Cadence na instanciação do indutor.

3. A partir dos valores conhecidos de I_{bias} , ω_0 e Q_L , estabelecer o valor de L para que V_{tanque} esteja na excursão mínima de saída. Sabe-se que V_{tanque} é:

$$V_{tanque} = I_{bias} \cdot \omega_0 \cdot L \cdot Q_L, \quad (6.3)$$

$$L = \frac{V_{tanque}}{I_{bias} \cdot \omega_0 \cdot Q_L} = \frac{550 \cdot 10^{-3}}{(2.2 \cdot 10^{-3}) \cdot (2 \cdot \pi \cdot 2.4375 \cdot 10^9) \cdot 7.8} = 2.09 \text{ nH}. \quad (6.4)$$

4. Calcular o valor de C do varactor a partir de L e da frequência central de oscilação do tanque LC. Sabe-se que ω_0 para um tanque ideal é dado por:

$$f = \frac{1}{2 \cdot \pi \cdot \sqrt{L \cdot C}}, \quad (6.5)$$

$$C = \frac{1}{(2 \cdot \pi \cdot 2.4375 \cdot 10^9)^2 \cdot 2.09276 \cdot 10^{-9}} = 2.04 \text{ pF}. \quad (6.6)$$

Onde os valores C_{MIN} e C_{MAX} são iguais a 1.89 pF e 2.19 pF, respectivamente.

5. Dado o mínimo ganho de malha fechada, $\alpha > 1$, calcular o valor da transcondutância mínima (g_m) para cada transistor NMOS. Sabe-se que g_m é igual a:

$$g_m = \alpha_{min} \cdot \frac{R \cdot C}{L}, \quad (6.7)$$

Onde R é considerada a resistência em série com o indutor, dada por:

$$R = \frac{\omega_0 \cdot L}{Q_L} = \frac{(2 \cdot \pi \cdot 2.4375 \cdot 10^9) \cdot (2.09276 \cdot 10^{-9})}{7.8} = 4.11 \text{ } \Omega, \quad (6.8)$$

Portanto, para α igual a 2, temos o valor da transcondutância igual a:

$$g_m = 2 \cdot \frac{4.10912 \cdot (2.03710 \cdot 10^{-12})}{2.09276 \cdot 10^{-9}} \approx 8 \text{ mS}. \quad (6.9)$$

6.3 Resultados do projeto elétrico do VCO

A topologia usada para o projeto está na Fig.(28). Observa-se que os resistores em série com os indutores foram retirados a fim de equiparar tal topologia a uma daquelas previamente estipuladas na Fig.(27). Além disso, nota-se que foram feitos vários ajustes nos valores dos componentes para que o resultado obtido fosse alcançado.

Os componentes usados são da biblioteca *tsmc18*, onde os transistores são *nmos2V* e *pmos2V*; os capacitores C_0 e C_1 são *moscap_rf*; C_2 *mimcap_2p0_wos*; e os indutores são *spiral_std_mu_x*.

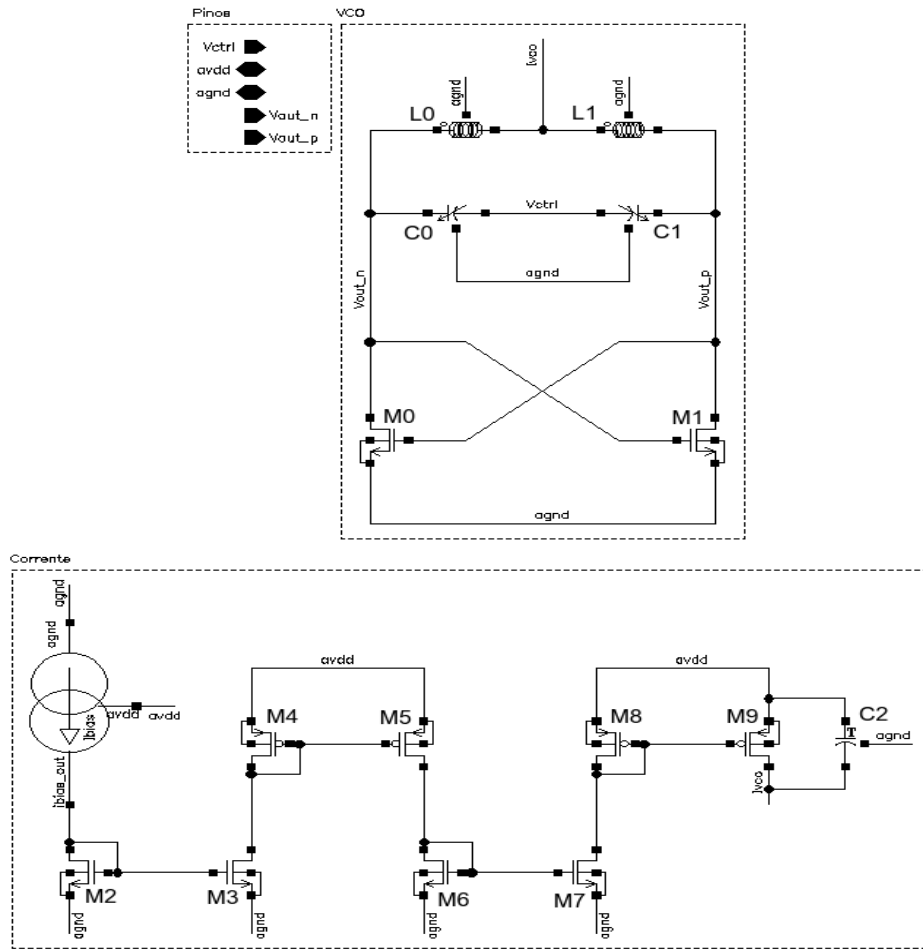


Figura 28 – Esquemático do VCO projetado.

- M0 e M1: $(W/L) = 200\mu\text{m}/8\mu\text{m}$, $fingers = 8$, $simM = 1$, $totalM = 8$;
- M2: $(W/L) = 700\text{nm}/2.8\mu\text{m}$, $fingers = 1$, $simM = 2$, $totalM = 2$;
- M3 e M4: $(W/L) = 2.8\mu\text{m}/2.8\mu\text{m}$, $fingers = 1$, $simM = 2$, $totalM = 2$;
- M5 e M6: $(W/L) = 5.6\mu\text{m}/2.8\mu\text{m}$, $fingers = 5$, $simM = 2$, $totalM = 10$;
- M7 e M8: $(W/L) = 61.6\mu\text{m}/2.8\mu\text{m}$, $fingers = 5$, $simM = 2$, $totalM = 10$;

- M9: $(W/L) = 61.6\mu\text{m}/2.8\mu\text{m}$, $fingers = 5$, $simM = 2$, $totalM = 10$;
- C0: $c = 4.1698\text{pF}$, $wf = 2.45\mu\text{m}$, $lf = 2.45\mu\text{m}$, $branch = 18$, $group = 6$;
- C1: $c = 4.1698\text{pF}$, $wf = 2.45\mu\text{m}$, $lf = 2.45\mu\text{m}$, $branch = 18$, $group = 6$;
- C2: $c = 871.224\text{fF}$, $lt = 21\mu\text{m}$, $wt = 21\mu\text{m}$, $lay = 6$;
- L0: $ind = 2.08989\text{nH}$, $w = 10\mu\text{m}$, $rad = 30\mu\text{m}$, $nr = 4$, $spacing = 2\mu\text{m}$, $lay = 6$;
- L1: $ind = 2.08989\text{nH}$, $w = 10\mu\text{m}$, $rad = 30\mu\text{m}$, $nr = 4$, $spacing = 2\mu\text{m}$, $lay = 6$.

A primeira parte da Fig.(28), corresponde à topologia do VCO tanque LC escolhida no Cap.(5). A segunda parte, refere-se aos espelhos de corrente implementados para gerar a corrente de 2.2 mA usada no VCO. Nota-se que a fonte de corrente de 5 μA não foi desenvolvida neste projeto, pode-se encontrar mais informações sobre ela no anexo(A).

A Fig.(29) apresenta o *testbench* utilizado para a simulação do VCO. Na sequência, estão dispostas diversas simulações importantes para a caracterização do projeto elétrico.

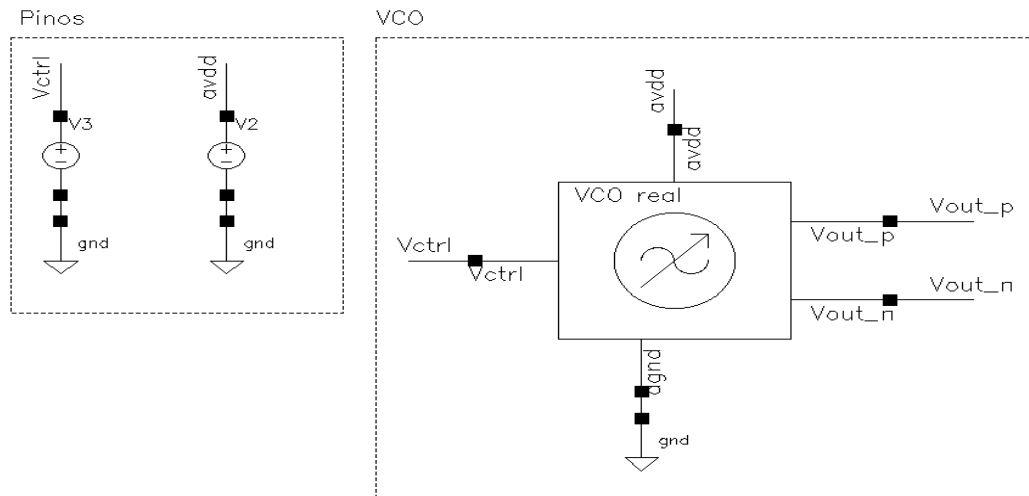


Figura 29 – *Testbench* do VCO projetado.

Percebe-se grandes diferenças entre as ondas de saída da simulação transiente do projeto elétrico e do modelo em alto nível. Uma vez que o modelo é ideal, este gera uma senoíde pura, com excursão máxima (0 a 1.8 V), centrada em 0.9 V e com saídas diferenciais em fase de 180°. Já no projeto elétrico, nota-se que as ondas não possuem um aspecto tão puro, a excursão de saída é significativamente menor (≈ 700 mV), o nível DC é mais baixo que o esperado e as saídas diferenciais possuem fase um pouco diferente de 180°. Tudo isto se deve às não idealidades dos componentes reais, principalmente do indutor. É válido frisar que na Fig.(31) são expostos parâmetros importantes das simulações, inclusive a estimativa de consumo de potência, 8.23683 mW.

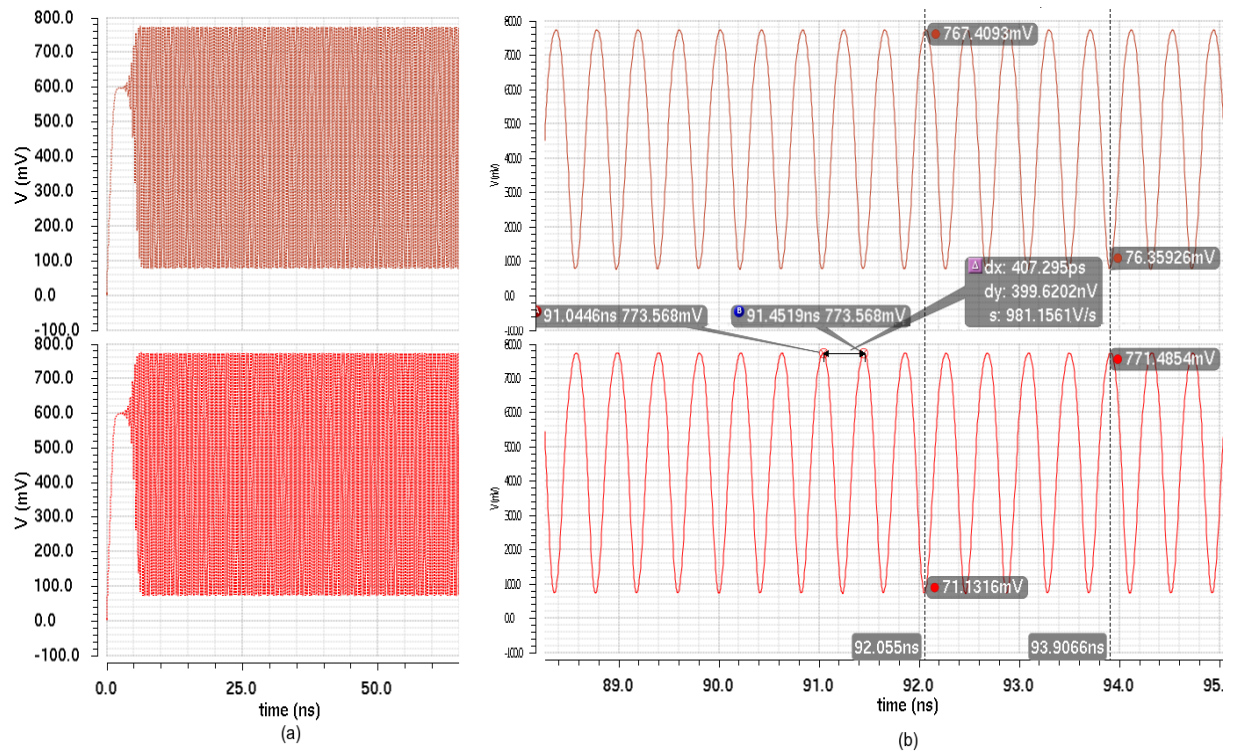


Figura 30 – Simulação transiente do VCO projetado: (a) sem *zoom*; (b) com *zoom*.

Outputs	
Name/Signal/Expr	Value
1 (Vctrl:CorFreq:PN-3dB)	(1.8 76390.0 -151.0)
2 Ibias	2.17317m
3 Freq_p	2.43615G
4 Freq_n	2.43615G
5 VoutN_average	460.457m
6 VoutP_average	468.691m
7 VoutP_swing	696.723m
8 VoutN_swing	702.565m
9 VoutN_min	71.1061m
10 VoutP_min	76.1138m
11 VoutP_max	772.837m
12 VoutN_max	773.671m
13 Power(W)	8.23683m

Figura 31 – ADE da simulação do projeto elétrico do VCO.

A simulação da Fig.(32) demonstra a resposta transiente do VCO para 16 tensões de entrada, é possível notar variações nas ondas quanto a excursão, nível DC e frequência, o que é de se esperar, pois a variação do V_{CTRL} muda a tensão no pino do varactor, o que altera sua capacitância, e como consequência, varia a impedância de saída dos estágios *single ended*, sendo estes as saídas do VCO. Portanto, os resultados obtidos nas simulações transientes, apesar das não idealidades, foram satisfatórios.

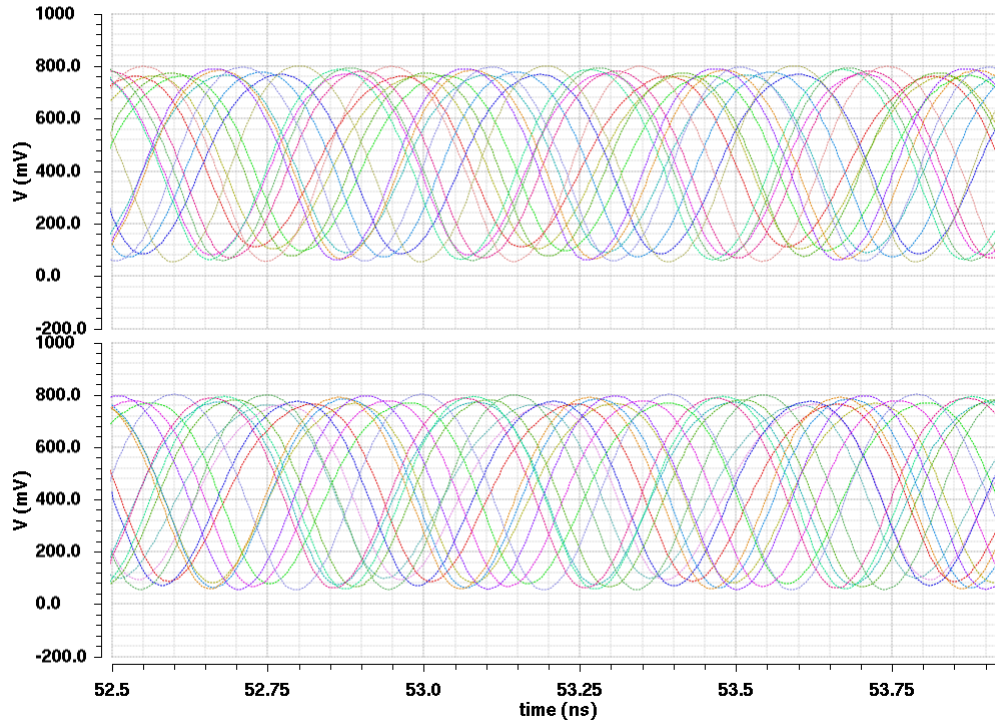


Figura 32 – Simulação paramétrica para 16 valores da tensão de controle.

As Fig.(33) e (34) são os resultados das simulações PSS e PNOISE, respectivamente. PSS é uma simulação que a partir da transiente procura alguma periodicidade no sinal analisado, sendo possível levantar as características de frequência de saída em relação à tensão de controle. Já a PNOISE é uma simulação que a partir de uma PSS, é capaz de estimar o ruído de fase de osciladores e outros circuitos de RF, através da densidade espectral de ruído devido aos componentes do circuito.

Na simulação PSS, pode-se inferir o comportamento do circuito com a variação da tensão de controle, de forma que é possível estimar o valor do ganho do VCO. Repara-se na Fig.(33)(a), a esquerda, que a variação da frequência com V_{CTRL} é não linear. A Fig.(33)(b), a direita, retrata a conduta dessa relação frequência *versus* V_{CTRL} na faixa de frequência estipulada para funcionamento do VCO, 2.35 GHz a 2.525 GHz. O ganho do bloco é dado pela derivada da função, no caso, uma aproximação seria a inclinação da reta que liga os pontos que passam pelos extremos da faixa de frequência de funcionamento do VCO, ou seja, 580.83 MHz/V.

A saída geradas pela simulação PNOISE é de extrema importância para o bom funcionamento do PLL, uma vez que as saídas do PLL são as próprias saídas do VCO. Segundo [Madureira \(2008\)](#), um parâmetro razoável para indicar a qualidade do oscilador é o ruído de fase a 1 MHz, este deve ser menor que -100 dBc/Hz. No caso, encontrou-se -106 dBc/Hz, considerado razoável. Outro ponto interessante é a forma da onda para valores próximos à portadora, a onda deve ser parecida com a apresentada no modelo de Leeson, Fig.(26), o que é verdade, observando na Fig.(34(b)).

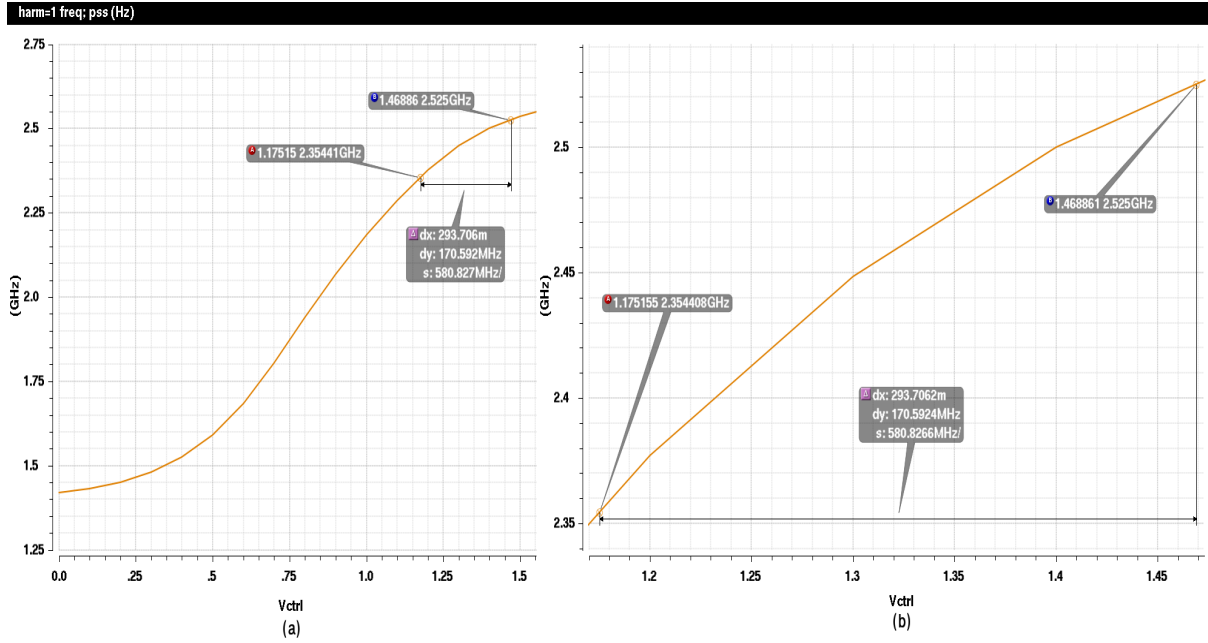


Figura 33 – Simulação PSS do VCO projetado: (a) sem *zoom*; (b) com *zoom*.

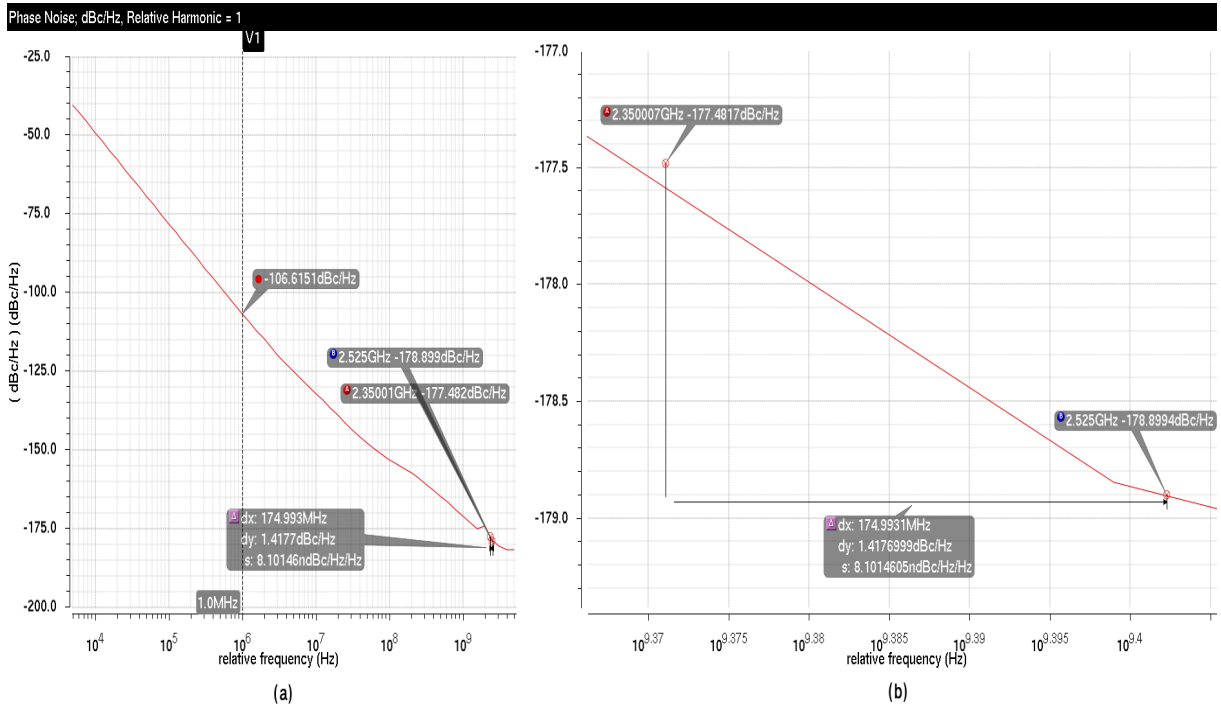


Figura 34 – Simulação PNOISE do VCO projetado: (a) sem *zoom*; (b) com *zoom*.

Por fim, é válido comentar uma das dificuldades enfrentadas neste projeto e necessidades para trabalhos futuros. Não foi possível modificar o valor do indutor nas ferramentas Cadence do laboratório da FGA, limitando bastante a obtenção de melhores resultados, principalmente quanto ao ruído. Portanto, vê-se a necessidade de otimizar o circuito elétrico do VCO, além de projetar o estágio *buffer* de saída e realizar as devidas validações de *corners* e Monte Carlo. No mais, os resultados são bastante razoáveis.

7 Projeto do PLL

O PLL é considerado um sistema único e complexo, desta forma é necessária uma abordagem especial, sendo importante observar as diferentes etapas do fluxo projeto a fim de garantir que cada passo adiante seja baseado nos parâmetros corretos. Abaixo, encontra-se um fluxograma que descreve, de forma simples, a metodologia de projeto seguida.

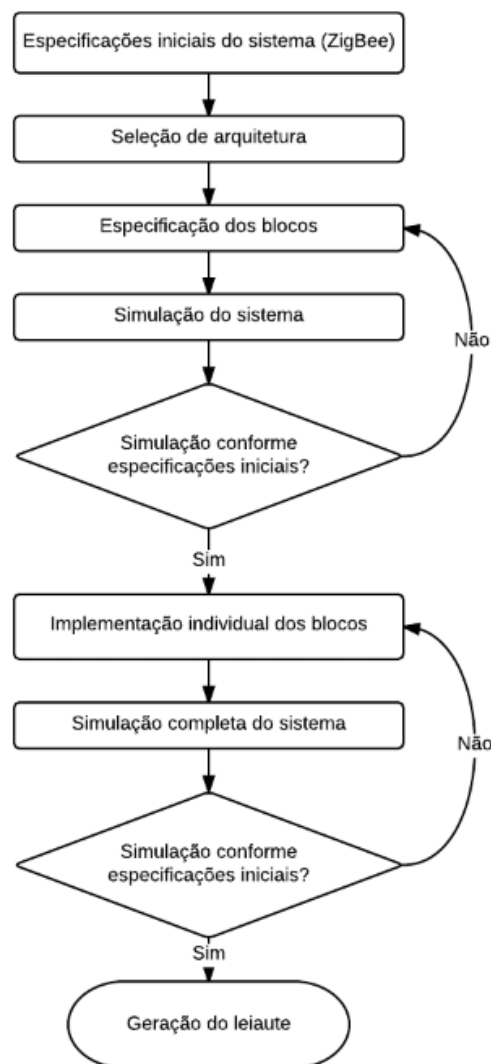


Figura 35 – Fluxo de projeto de um PLL [Bistue, Quemada e Adin (2009)].

O primeiro passo a ser dado é a determinação inicial das especificações do sintetizador, algumas são extraídas diretamente do padrão da aplicação, no caso do ZigBee, conforme a Tab.(3). O passo 2 é a escolha da arquitetura, levando em consideração as vantagens e desvantagens de cada uma para a aplicação em questão. Em seguida, determina-se os requisitos de cada bloco, para este passo é útil usar simultaneamente fontes providas de experiência de projeto de projetistas, publicações científicas e *datasheets*.

Após o levantamento de dados do sistema e dos blocos individuais, é essencial o uso de programas de simulação, criando-se modelos executáveis do sistema em alto nível. Esta abordagem oferece a vantagem do projetista ter total controle do número de variáveis que é introduzida tanto no processo quanto no modelo utilizado.

Gerando-se a simulação, caso as especificações do sistema não tenham sido alcançadas, volta-se no passo referente a determinação das especificações dos blocos, ficando neste laço até que a simulação gere os resultados esperados. Quando a simulação em alto nível estiver de acordo com as especificações do sistema, é possível iniciar o projeto de cada bloco separadamente com alto nível de detalhe, utilizando sempre uma ferramenta de simulação, a fim de atingir os requisitos dos blocos, levantados anteriormente. Observa-se que neste ponto é inevitável a interação entre os projetistas e os gerentes de sistema, desempenhando a troca de informações entre as partes para otimização global do sistema.

Com os blocos projetados individualmente em baixo nível de abstração, a simulação do sistema é novamente realizada, entrando-se num laço com o passo da implementação dos blocos individuais, caso a simulação não ocorra como esperado. Se o resultado da simulação for compatível com as especificações do sistema, o fluxo de projeto termina e o leiaute é gerado.

7.1 Especificações do Sintetizador

A Tabela (3) introduz os dados fundamentais da camada PHY do padrão da aplicação, IEEE 802.15.4.

Tabela 3 – Especificações 2450 MHz IEEE 802.15.4 camada PHY [Oh e Lee (2006)].

Métricas de performance	Especificações
Portadora	2400 MHz
Espectro	2400-2483.5 MHz
Modulação	O-QPSK usando DSSS
Espaço do canal	5 MHz
Número de canais	16 (11-26 na camada PHY)
Sensibilidade	-85 dBm
SNR	2 dB
<i>Settling Accuracy</i>	+/- 40 ppm (96 KHz)
<i>Alternate Channel Rejection</i>	30 dB a 10 MHz <i>offset</i>
<i>Adjacent Channel Rejection</i>	0 dB a 5 MHz <i>offset</i>
Potência de transmissão de saída	-5 a 3 dBm
Taxa de transmissão de dados	250 Kb/s

7.2 Projeto a Nível de Sistema

A topologia proposta foi a de um PLL N-inteiro com saídas analógicas diferenciais. O diagrama de blocos da topologia a ser utilizada encontra-se na Fig.(36).

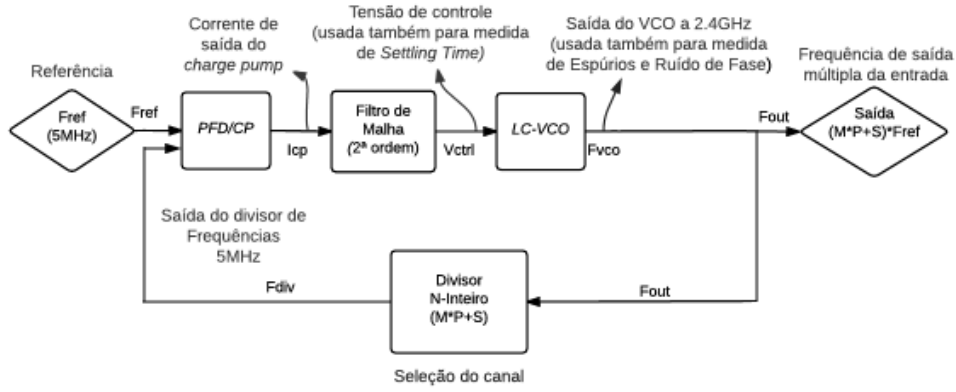


Figura 36 – Diagrama de blocos da topologia de PLL proposta.

As equações importantes para o projeto deste PLL estão na sequência. O procedimento completo de dedução das mesmas será omitido a fim de objetividade, entretanto podem ser encontrados em Razavi e Behzad (1998) e Srinivasan (2006). O ganho em malha aberta, neste circuito é:

$$G(s) = \frac{K_{VCO} \cdot I_{CP}}{2\pi N C_1} \cdot \frac{(1 + \frac{s}{\omega_{z1}})}{s^2 \cdot (1 + \frac{s}{\omega_{p1}})}, \quad (7.1)$$

onde $\omega_{z1} = \frac{1}{R_1 \cdot C_1}$ e $\omega_{p1} = \frac{1}{R_1 \cdot C_2}$.

O ganho em malha fechada (4.3), com as devidas reduções e rearranjos, para que tenha a forma de um clássico sistema de segunda ordem, é dado pela Eq.(7.2).

$$F(s) = \frac{\omega_n^2}{s^2 + (2\zeta\omega_n) \cdot s + \omega_n^2}, \quad (7.2)$$

onde $\omega_n = \sqrt{\frac{K_{VCO} \cdot I_{CP}}{2\pi N C_1}}$, $\omega_c = (2\zeta\omega_n) = \frac{K_{VCO} \cdot I_{CP} \cdot R_1}{2\pi N}$ e $\zeta = \frac{R_1}{2} \cdot \sqrt{\frac{K_{VCO} \cdot I_{CP} \cdot C_1}{2\pi N}}$.

Pela Fig.(38) é possível ver que as localizações do polo ω_{p1} e do zero ω_{z1} , afetam a dinâmica do *loop*. Por Ogata e Yang (1970), tem-se:

$$\omega_{z1} = \frac{\omega_c}{\alpha^2} \quad (7.3)$$

$$\omega_{p1} = \omega_c \cdot \alpha^2 \quad (7.4)$$

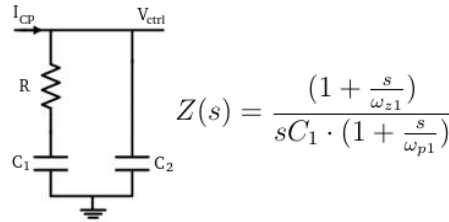


Figura 37 – Filtro de malha de segunda ordem e a sua função de transferência [Srinivasan (2006)].

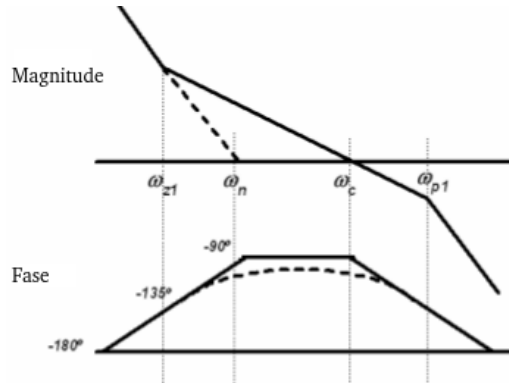


Figura 38 – Localização dos polos e zeros [Srinivasan (2006)].

Assumindo o caso de amortecimento crítico para um sistema de segunda ordem, $\alpha = 2$ e $\zeta = 1$. Este cenário leva a um bom compromisso entre *settling time* e *overshoot*.

7.3 Procedimento de Projeto

O projeto contará com a tecnologia TSMC 0.18 μm CMOS e usará tensão de referência igual a 1.8V.

Passo 1: Extrair os parâmetros necessários a partir do padrão da aplicação utilizada.

O protocolo ZigBee apresenta faixa de operação entre 2400 MHz e 2483.5 MHz, possui 16 canais, distribuídos entre o 11 e o 26 da camada PHY, com 5 MHz de espaçamento entre si, de 2400 MHz a 2475 MHz. Desta forma, o sintetizador deve possibilitar a seleção destes 16 canais com precisão de frequência de 40-ppm. A partir destas informações é possível justificar o uso da arquitetura proposta, devido a relação de divisão necessária, sendo natural a escolha de uma frequência de referência de 5 MHz, coincidindo com o espaço entre os canais, em conjunto com a arquitetura N-inteiro.

- F_{CANAL} : $2400 + 5 (k - 11)$ MHz, onde $k = 11$ a 26;
- F_{REF} : 5 MHz;
- F_{OUT} : 2400 a 2475 MHz;

- F_{VCO} : 2400 a 2475 MHz;
- $F_{CENTRALVCO}$: 2437.5 MHz;
- F_{DIV} : é a frequência comparada ao F_{REF} no PFD para seleção de canais;
- Fator de divisão (N): 480 a 495, resultado da divisão de 2400 e 2475 pelo espaçamento dos canais, 5 MHz, respectivamente;

É possível caracterizar os dados do bloco divisor, composto pelos sub-blocos: *Dual Modulus* (M), *Swallow Counter* (S) e *Programmable Counter* (P). Este bloco é regido pela Eq.(7.5):

$$F_{out} = (M \cdot P + S) \cdot F_{div}. \quad (7.5)$$

Para o valor de $M = 15$ e sabendo que o valor de $S = 16$, devido a quantidade de canais, caracteriza-se um divisor 15/16 onde $P = 32$.

Neste ponto, define-se o valor máximo aceitável de ruído de fase e rejeição de espúrios da saída do VCO, a partir de dados oriundos da Tabela (3). Estes parâmetros são limitados pelos dados de *adjacent channel rejection* e rejeição de canal adjacente. Segundo Srinivasan (2006), para ruído de fase (7.6) e (7.7) e rejeição de espúrios (7.8), temos:

$$P_{sig} + P_{LO} - (P_{interferer} + PN + P_{BW}) > SNR_{min}, \quad (7.6)$$

$$PN - P_{LO} < (P_{sig} - P_{interferer}) - P_{BW} - SNR_{min} \quad e \quad (7.7)$$

$$P_{sp} - P_{LO} < (P_{sig} - P_{interferer}) - SNR_{min}. \quad (7.8)$$

Onde:

- P_{sig} : potência do conteúdo da portadora;
- P_{LO} : potência do conteúdo do LO;
- PN: contribuição do LO ao ruído de fase;
- PBW: potência do conteúdo do sinal por toda a largura do canal;
- $P_{interferer}$: potência do conteúdo da interferência;
- P_{sp} : contribuição do LO para emissão de sinais espúrios ;
- SNR_{MIN} : relação sinal ruído mínima na entrada da seção IF seguinte ao *downconversion* para demodulação e taxa de erro de *bits* tolerável.

Pela Tabela (3) temos que $P_{interferer}$ a 5 MHz é igual a 0 dB e $P_{interferer}$ a 10 MHz é igual a 30 dB. Por Srinivasan (2006), $SNR_{MIN} = 8$ dB e $P_{sig} = 0$. Assumindo a largura de banda de 5 MHz, temos:

$$(PN - P_{LO})_{5 \text{ MHz}} < (0 - 0) - 10 \cdot \log(5 \cdot 10^6) - 8 = -75 \text{ dBc/Hz} \quad e \quad (7.9)$$

$$(PN - P_{LO})_{10 \text{ MHz}} < (0 - 30) - 10 \cdot \log(5 \cdot 10^6) - 8 = -105 \text{ dBc/Hz}. \quad (7.10)$$

Para uma margem de 5 dB devido a não linearidades do sistema, a especificação do ruído de fase é -80 dBc/Hz para uma frequência de *offset* a 5 MHz da portadora e -110 dBc/Hz para 10 MHz.

$$(P_{sp} - P_{LO})_{5 \text{ MHz}} < (0 - 0) - 8 = -8 \text{ dBc/Hz} \quad e \quad (7.11)$$

$$(P_{sp} - P_{LO})_{10 \text{ MHz}} < (0 - 30) - 8 = -38 \text{ dBc/Hz}. \quad (7.12)$$

Novamente, para uma margem de 5 dB, a especificação de rejeição de espúrios para frequências de *offset* a 5 MHz e 10 MHz da portadora é -13 dBc/Hz e -43 dBc/Hz, respectivamente.

Passo 2: Estabilidade do *loop*.

Para o procedimento, a priori, necessita-se assumir a corrente do *charge pump* e o ganho do VCO (K_{VCO}). Estipula-se a corrente do *charge pump* como 20 μ A, baseado em Srinivasan (2006) e Ismail e Othman (2009). Para o ganho do VCO, este é dado pela razão entre a faixa de operação do VCO e a faixa de valores da tensão de controle. Para uma faixa de valores da tensão de controle de 1 V, o ganho do VCO é dado por:

$$K_{VCO} = \frac{(2475 - 2400) \text{ MHz}}{1 \text{ V}} = 75 \text{ MHz/V}. \quad (7.13)$$

A frequência natural do *loop* deve ser significativamente menor que a frequência de referência, de modo que os componentes de ruído e espúrios da frequência de referência sejam atenuados pelo filtro de malha. Por Ismail e Othman (2009), assume-se:

$$\omega_n < \frac{\omega_{ref}}{10}. \quad (7.14)$$

Por Shu e Sánchez-Sinencio (2006) e Ismail e Othman (2009), para PLLs de segunda ordem, a relação entre frequência natural e *settling time* é:

$$\omega_n > \frac{2\pi}{t_{lock}}. \quad (7.15)$$

Portanto, uma faixa de prováveis valores para a frequência natural é estabelecida por (7.16). Uma vez que *settling time* e largura de banda de *loop* são diretamente afetados pelas ω_n , há conflitos de interesse, então compromissos devem ser feitos:

$$\frac{2\pi}{t_{lock}} < \omega_n < \frac{\omega_{ref}}{10}. \quad (7.16)$$

Para $t_{lock}=192 \mu s$, máximo da aplicação [Instruments (2006)], temos:

$$32.72 \text{ Krad/s} < \omega_n < 3.14 \text{ Mrad/s}. \quad (7.17)$$

Segundo Ogata e Yang (1970), a relação entre a frequência natural com a largura de banda do *loop* é dado por:

$$\omega_n = \frac{\omega_c}{2\zeta}. \quad (7.18)$$

Escolhendo uma frequência natural de 49.35 KHz (310.10 Krad/s), o que gera valores razoáveis para os componentes, e usando fator de amortecimento crítico 1, tem-se:

$$\omega_c = 2 \cdot 1 \cdot (2 \cdot \pi \cdot 49.35) \text{ Krad/s} \approx 620.20 \text{ Krad/s}. \quad (7.19)$$

Considerando $\alpha = 2$ e $\zeta = 1$, através das Eqs. (7.3) e (7.4):

$$\omega_{z1} = \frac{\omega_c}{2^2} \approx 155.05 \text{ Krad/s} \quad e \quad (7.20)$$

$$\omega_{p1} = \omega_c \cdot 2^2 \approx 2480.8 \text{ Krad/s}. \quad (7.21)$$

Portanto com estes dados, a margem de fase esperada calculada a seguir:

$$PM = \tan^{-1} \left(\frac{\omega_c}{\omega_{z1}} \right) - \tan^{-1} \left(\frac{\omega_c}{\omega_{p1}} \right) = \tan^{-1}(4) - \tan^{-1} \left(\frac{1}{4} \right) \approx 61.93^\circ. \quad (7.22)$$

Passo 3: Parâmetros do filtro de malha.

Após definir os valores da frequência natural (ω_n), polo (ω_{p1}) e do zero (ω_{z1}), pode-se calcular os parâmetros do filtro de malha de segunda ordem, Fig.(37), ou seja, os valores dos capacitores e resistor.

$$\omega_n = \sqrt[2]{\frac{K_{VCO} \cdot I_{CP}}{2\pi N C_1}}. \quad (7.23)$$

Isolando C_1 e substituindo os valores, onde N é considerado como o dobro da média daqueles achados anteriormente (480 e 495), ou seja, $N = 975$, tem-se:

$$C_1 = \frac{K_{VCO} \cdot I_{CP}}{2\pi N \omega_n^2} = \frac{(2\pi 75 \cdot 10^6) \times 20 \cdot 10^{-6}}{2\pi (975) \cdot (310.1 \cdot 10^3)^2} \approx 16 \text{ pF}. \quad (7.24)$$

Através das expressões do polo e do zero é possível deduzir os valores de R_1 e C_2 :

$$R_1 = \frac{1}{\omega_{z1}C_1} = \frac{1}{(155.05 \cdot 10^3) \times (16 \cdot 10^{-12})} \approx 403.13 \text{ K}\Omega \text{ e} \quad (7.25)$$

$$C_2 = \frac{1}{\omega_{p1}R_1} = \frac{1}{(2480.8 \cdot 10^3) \times (403.13 \cdot 10^3)} \approx 1 \text{ pF}. \quad (7.26)$$

Passo 4: *Settling Time*;

Por [Valero-Lopez \(2004\)](#) e [Moon \(2005\)](#), para o caso de sistema criticamente amortecido, o *settling time* é dado por:

$$t_{lock} = \frac{1}{\zeta\omega_n} \cdot \ln\left(\frac{\Delta f}{\alpha f_0}\right), \quad (7.27)$$

onde α é acurácia, de acordo com o protocolo ZigBee, aproximadamente 40 ppm, Δf é a faixa de frequência máxima de sintonização 75 MHz e f_0 a frequência da portadora, média entre 2475 MHz e 2400 MHz, 2437.5 MHz. Portanto, t_{lock} , através desta aproximação, é por volta de 19.60 μ s, 10.21% do pior caso especificado para aplicação.

Passo 5: Verificação a nível de sistema.

Para tal, desenvolveu-se uma função no MATLAB ([D.1](#)) onde a partir de determinados parâmetros de entrada, obtêm-se os dados dos passos 1-4 ([98](#)) e os diagramas de bode da resposta em malha aberta e fechada, [Fig.\(39\)](#).

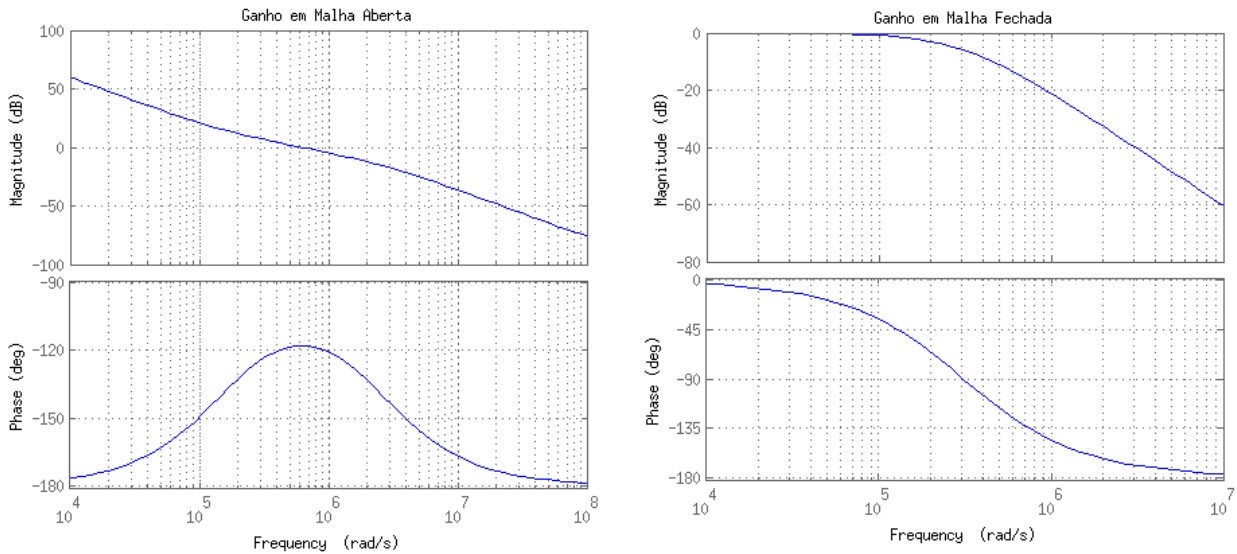


Figura 39 – Gráfico de Bode da resposta em malha aberta e malha fechada, respectivamente.

A [Fig.\(40\)](#) mostra o comportamento das funções de transferência em malha aberta e fechada com a variação da frequência natural.

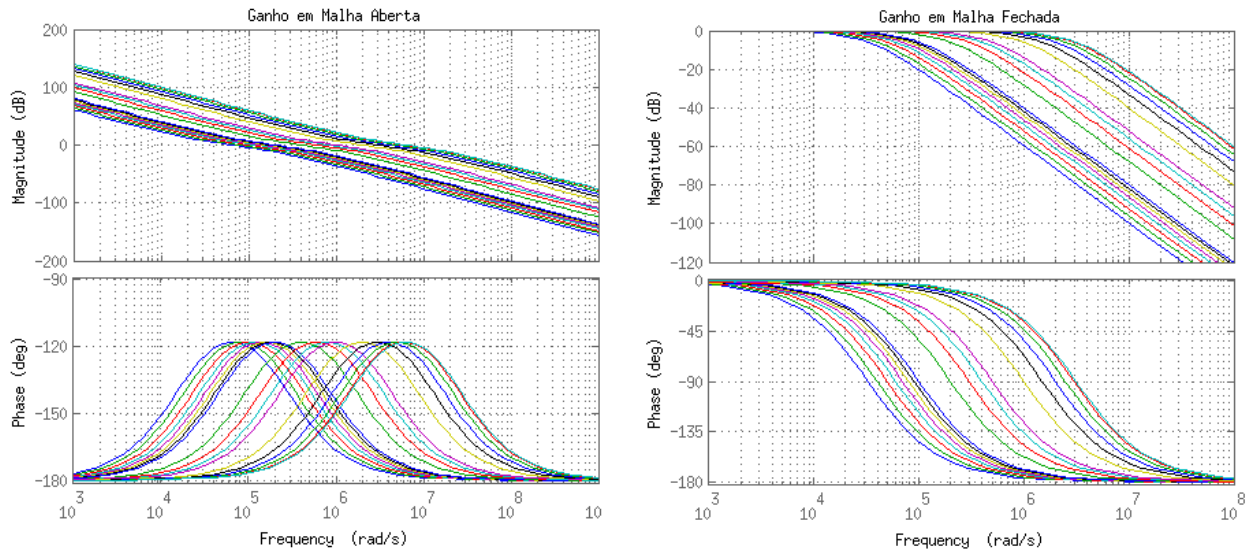


Figura 40 – Variação da resposta em malha aberta e fechada, respectivamente, com ω_n .

O gráfico da Fig.(41) e a Tab.(4) ilustram a relação entre os valores dos componentes do filtro de malha com a variação da frequência natural. Observa-se que com o aumento da frequência os valores dos capacitores C_1 e C_2 diminuem, enquanto o valor do resistor R_1 aumenta. É preciso dar atenção a estas grandezas, uma vez que capacitores e resistores grandes ocupam muita área de *chip* e inserem quantidades consideráveis de ruído ao sistema. Na Tab.(4), selecionou-se valores dos componentes em uma faixa da frequência natural que pode ser interessante para a modelagem, de forma que gera uma resposta rápida, com largura de banda razoável e componentes de tamanho aceitável.

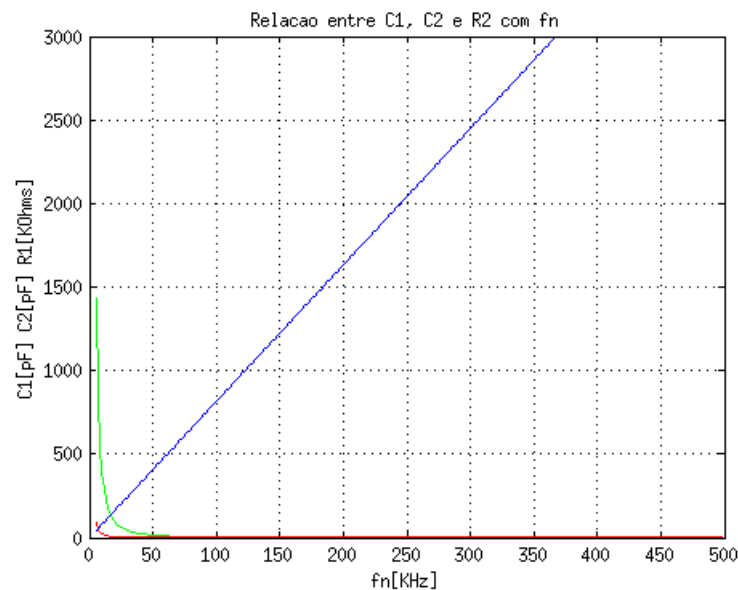


Figura 41 – Variação dos componentes do filtro de malha com f_c .

Tabela 4 – Parâmetros do filtro de malha com a variação de ω_n .

Frequência Natural [Krad/s] ou [KHz]	Largura de Banda [Krad/s] ou [KHz]	C_1 [pF]	C_2 [pF]	R_1 [K Ω]
100.00 ou 15.92	200.00 ou 31.83	153.85	9.62	130.00
200.00 ou 31.83	400.00 ou 63.66	38.46	2.40	260.00
300.00 ou 47.75	600.00 ou 95.49	17.09	1.07	390.00
310.10 ou 49.35	620.20 ou 98.68	16.00	1.00	403.13
340.00 ou 54.11	680.00 ou 108.23	13.31	0.83	442.00
360.00 ou 57.30	720.00 ou 114.59	11.87	0.74	468.00
380.00 ou 60.48	760.00 ou 120.96	10.65	0.67	494.00
400.00 ou 63.66	800.00 ou 127.32	9.62	0.60	520.00
500.00 ou 79.58	1000.00 ou 159.15	6.15	0.38	650.00
600.00 ou 95.49	1200.00 ou 190.99	4.27	0.27	780.00

Um compromisso entre largura de banda, *settling time*, ruído de fase, emissão de sinais espúrios, consumo de potência e espaço em área *chip*, é extremamente necessário para cumprimento das especificações iniciais de projeto. Portanto:

- Se o interesse é baixo ruído de fase, deve-se atentar principalmente ao ruído produzido pelo VCO;
- Se o interesse é baixa emissão de sinais espúrios, é preciso atentar a fonte destes sinais, como o *charge pump* e o filtro de malha;
- Se o interesse é garantir uma boa área de *chip*, os componentes do filtro de malha e do VCO devem ser inspecionados;
- Se o interesse é relacionado a largura de banda e *settling time*, o bloco de interesse é o filtro de malha.

8 Planejamento da Modelagem do PLL

Como dito anteriormente, será projetado um PLL com saídas diferenciais analógicas cuja faixa de frequência de operação vai de 2400 MHz a 2475 MHz. Neste ponto vale especificar detalhadamente os blocos a serem modelados e o tipo de sinal que cada um irá receber e gerar. Abaixo, encontra-se um diagrama de blocos que ilustra tais especificações.

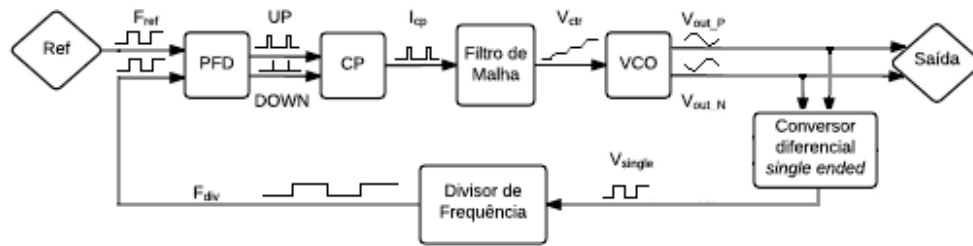


Figura 42 – Diagrama de blocos do PLL e tipos de sinais.

Observa-se que neste diagrama foi acrescentado o bloco referente ao conversor de saída diferencial para *single ended*, uma vez que o VCO tanque LC modelado possui saída diferencial, pelos motivos descritos anteriormente. Desta forma, este bloco é responsável pela conversão das duas saídas do bloco VCO para uma única saída, usada na realimentação. Outra função do conversor de saída diferencial para *single ended* é a conversão dos sinais analógicos da saída do VCO em digital.

No diagrama da Fig.(42) também é possível distinguir os tipos de sinais de entrada e saída de cada bloco. Partindo do fato que o sinal de referência (*clock*) é um sinal digital, o PFD recebe como entrada dois sinais digitais, F_{ref} e F_{div} , e gera como saídas 2 sinais, *UP* e *DOWN*, digitais. O CP recebe como entrada as duas saídas digitais do PFD e gera em sua saída pulsos de corrente. O filtro de malha, por sua vez, recebe os pulsos de corrente do CP e gera um nível de tensão DC. Já o VCO, recebe o nível de tensão DC do filtro de malha e gera duas saídas analógicas, onde estas saídas são as saídas do PLL, de forma que, para a realimentação, o conversor de saída diferencial para *single ended* recebe as mesmas e as transforma em uma única saída digital. Por fim, o divisor de frequência, recebe a saída do conversor como entrada e gera uma única saída digital, F_{div} , utilizada como entrada no PFD.

Sabendo os blocos a serem modelados e os tipos de sinais que estes devem receber e gerar, pode-se efetuar a modelagem do sistema. Primeiramente, é necessário seguir os passos da metodologia de projeto do PLL descritos no Cap.(7), representados pelo diagrama da Fig.(43). Sendo assim, com os resultados obtidos no capítulo em questão, tem-se um ponto inicial para a modelagem.

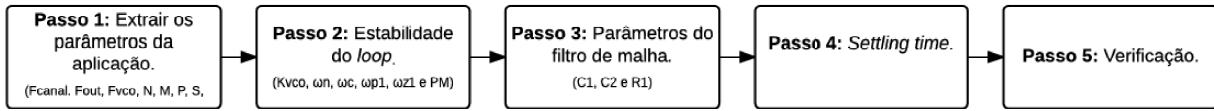


Figura 43 – Passos da Metodologia de Projeto do PLL.

Entretanto, nos passos da metodologia de projeto do PLL efetuados no Cap.(7), considerou-se o sinal de saída do VCO com uma frequência de operação entre 2.4 GHz e 2.75 GHz, de forma que, nesta configuração, a tensão de controle do VCO para os canais 11 e 26, extremos, possui comportamento indesejado, onde a saída do PLL converge mesmo se a tensão de controle do VCO divergir. Por este motivo, a faixa de frequência de operação do VCO foi esticada para 2.35 GHz a 2.525 GHz, portanto, o ganho do VCO, K_{VCO} , foi para 175 KHz/V e a partir deste novo valor, pode-se refazer os passos da metodologia de projeto.

Considerando o novo K_{VCO} , os parâmetros do filtro de malha serão alterados, logo, para que estes não fiquem com valores destoantes aos reais, a frequência natural será modificada. Segundo o código desenvolvido em MATLAB (D.1) e Fig.(99), considerando a alteração do K_{VCO} , uma boa alternativa para o valor desta frequência é 473.700 Krad/s (ou 75.39 KHz), onde temos: $C_1 = 16$ pF, $C_2 = 1$ pF, $R = 263.92$ K Ω (usou-se $R = 265$ K Ω a fim de melhor aproximar a valores usuais) e $t_{lock} = 14.47$ μ s, 7.54% dos 192 μ s permitidos.

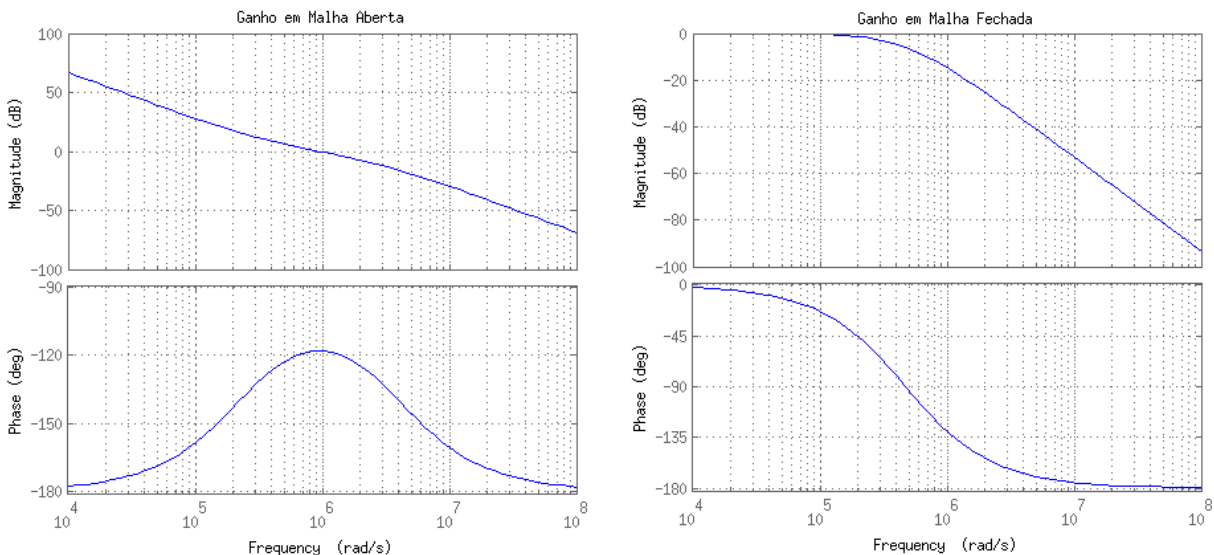


Figura 44 – Gráfico de Bode da resposta em malha aberta e malha fechada, respectivamente.

9 Modelagem do PLL

A modelagem do PLL foi feita a partir das especificações já expostas. Na sequência, cada um dos blocos pertencentes ao sistema serão modelados e apresentados a partir de sua descrição funcional, tabela de pinos e simulação. Observa-se que a modelagem foi feita com um nível de abstração mais baixo, uma vez que os componentes internos de cada bloco do sistema foram descritos, testados e posteriormente unidos para formação da hierarquia superior, validando assim as topologias escolhidas. Os demais blocos importantes para a composição do sistema estão no apêndice.

A Tab.(5) é composta por alguns dos parâmetros que são comuns à maioria dos blocos modelados. Nota-se que a alimentação dos blocos é de 1.8V, todos os blocos possuem a entrada referente ao *enable* para habilitar/desabilitar, a tensão de referência para transições (v_{th}) é metade de vdd (0.9V) e os tempos de transição t_f e *delay* t_d dos modelos são de 1 fs, a fim de estabelecer um modelo mais próximo ao ideal.

Tabela 5 – Parâmetros de simulação comuns aos blocos.

Parâmetro	Descrição	Valor
avdd	Tensão de alimentação	1.8 V
agnd	Tensão de referência (<i>ground</i>)	0 V
en	Tensão de <i>enable</i>	0 V
v_{th}	Tensão de referência de transição de estados do modelo	0.9 V
t_f	Tempo de transição de subida e descida do modelo	1 fs
t_d	Tempo de atraso do modelo	1 fs

9.1 Detector de Fase e Frequência

O PFD possui a funcionalidade de transformar a diferença de fase e frequência de suas duas entradas em dois sinais, *UP* e *DOWN*, proporcionais a essa diferença. A concepção deste bloco foi feita a partir de duas abordagens, codificando-o diretamente em Verilog-AMS e a partir da modelagem individual de cada bloco interno, de acordo com a topologia da Fig.(11-a). Esta última, mostrou-se ineficiente devido a presença de realimentação, tentou-se utilizar *buffers* para inserção do atraso intrínseco às portas lógicas, entretanto não se obteve êxito. Para maiores informações do bloco e sua implementação a nível de transistores, [Gomes \(2015\)](#).

9.1.1 Descrição do Bloco

A descrição em Verilog-AMS deste bloco parte da implementação dos *flip flops*, da porta AND e da realimentação existente entre eles. Tal comportamento é obtido através do uso da função *cross*, em relação a borda de subida da entrada em questão, em conjunto com a detecção da borda de subida da entrada *reset*. Desta forma, a saída *UP* vai para nível alto quando há diferença de fase entre f_{ref} e f_{div} e a saída *DOWN* quando há diferença entre *UP* e f_{div} , correspondendo com a dinâmica do circuito da Fig.(45).

```

1  analog begin: main
2      enable = (V(en) > vth) ? 0 : 1;
3
4      @(initial_step) begin
5          dig1 = 0; dig2 = 0;
6      end
7
8      @(cross(V(fref) - vth, 1) or posedge(reset));
9          reset = dig1 && dig2; dig1 = (V(fref) > vth);
10
11      @(cross(V(fdiv) - vth, 1) or posedge(reset));
12          reset = dig1 && dig2; dig2 = (V(fdiv) > vth) && (dig1 > vth);
13
14      V(UP) <+ transition(V(avdd)*dig1*!reset*(enable), td, tt);
15      V(DOWN) <+ transition(V(avdd)*dig2*!reset*(enable), td, tt);
16  end

```

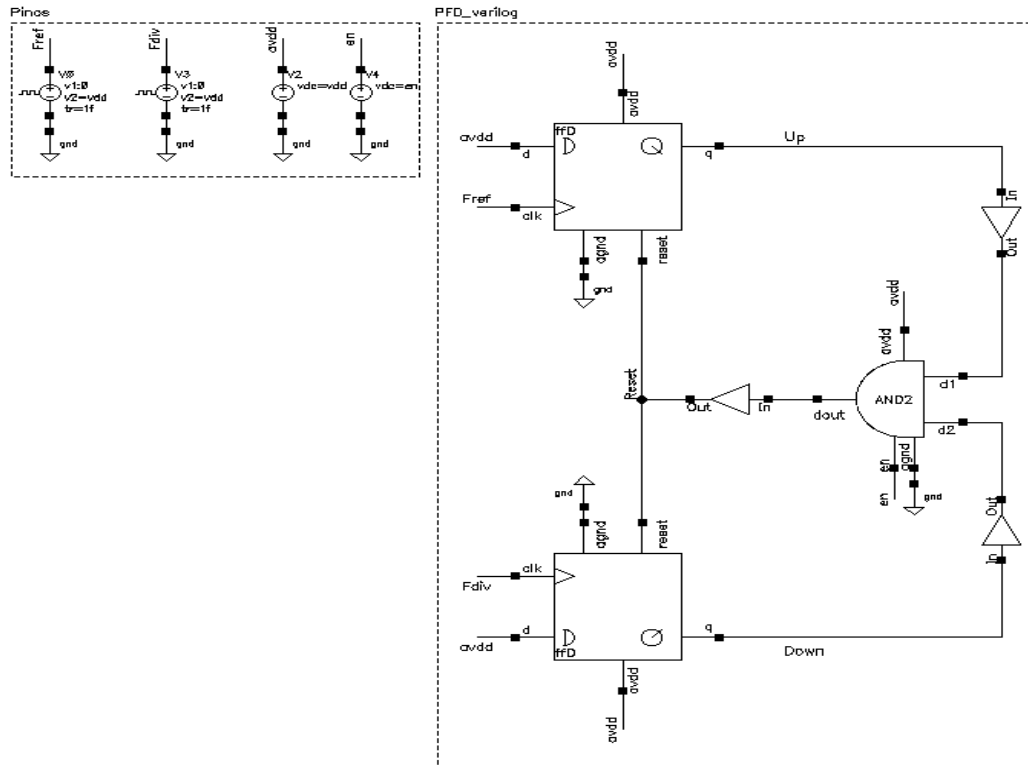


Figura 45 – Esquemático do PFD_v1.

9.1.2 Descrição dos Pinos

Tabela 6 – Descrição dos pinos do detector de fase e frequência.

Pino	Descrição	Tipo
F_{ref}	Sinal de referência	<i>input</i>
F_{div}	Sinal da realimentação proveniente do divisor	<i>input</i>
up	Saída do <i>flip flop</i> D que recebe F_{ref} como <i>clock</i>	<i>output</i>
down	Saída do <i>flip flop</i> D que recebe F_{div} como <i>clock</i>	<i>output</i>

9.1.3 Simulação

A simulação e os parâmetros configurados são apresentados a seguir, observa-se que foi usado o mesmo valor de período entre as entradas e inseriu-se um pequeno *delay*, diferença de fase, entre elas. Este cenário é razoável no ponto de vista do comportamento esperado do PLL, onde na realimentação, o sinal f_{div} terá frequência próxima ao sinal de referência com uma pequena diferença de fase. As ondas de saída, Fig.(47), estão de acordo com o esperado, validando-se assim a modelagem deste bloco.

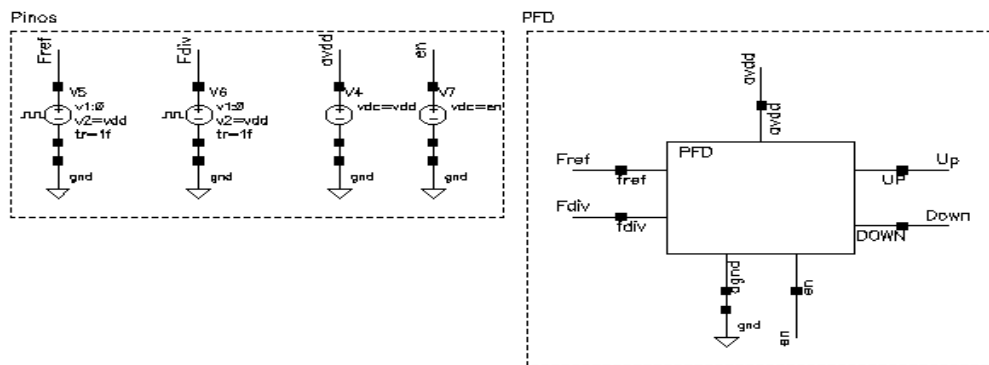


Figura 46 – Testbench do PFD.

Tabela 7 – Parâmetros de simulação do detector de fase e frequência.

Parâmetro	Descrição	Valor
periodo	Período de operação das fontes de teste	(1/5 MHz) s
<i>delay</i>	Tempo de atraso entre as fontes de teste	25 ns
ciclos	Quantidade de ciclos usado na simulação	10

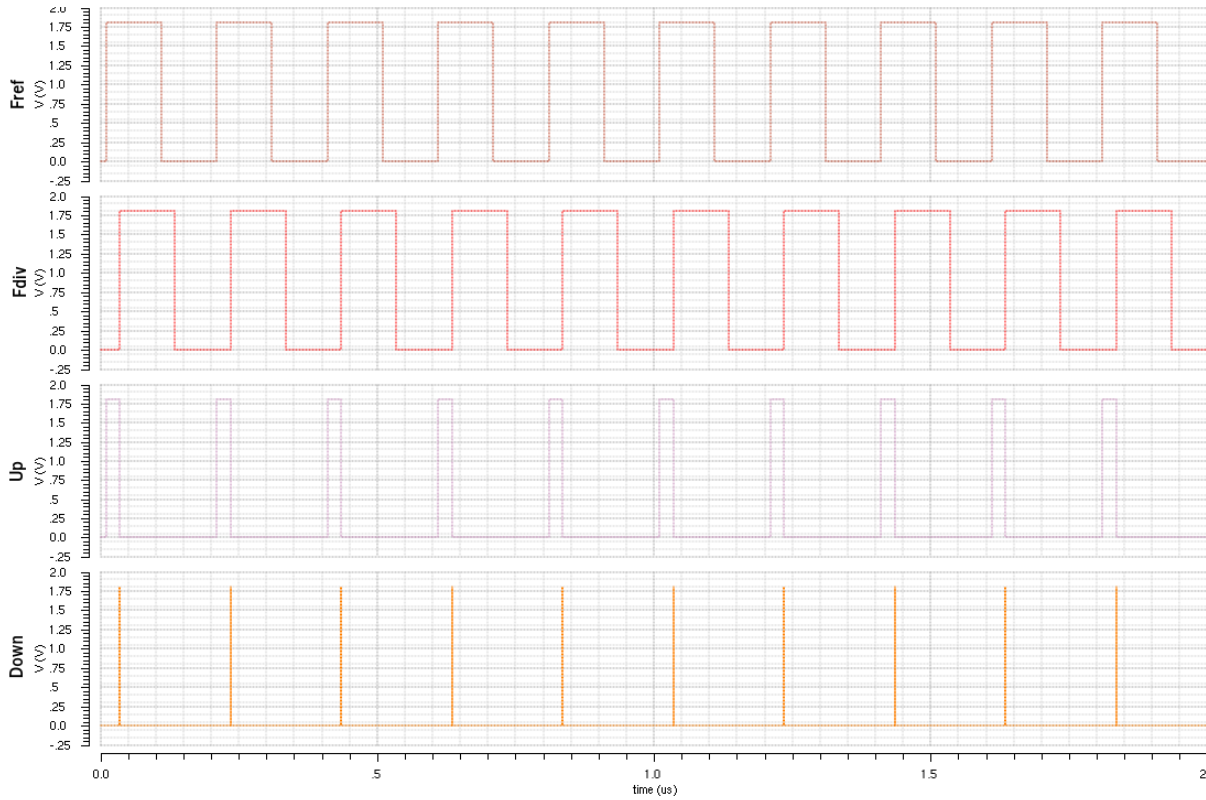


Figura 47 – Simulação do PFD.

9.2 Charge Pump

O *charge pump* é responsável por gerar pulsos de corrente proporcionais às saídas *UP* e *DOWN* do PFD, desta forma, quando *UP* está em nível alto, insere-se um montante de corrente, e quando *DOWN* se encontra em alto, retira-se o mesmo montante. A modelagem deste bloco foi feita em Verilog-AMS, conforme o código a seguir. Para maiores informações do bloco e sua implementação a nível de transistores, [Gomes \(2015\)](#).

9.2.1 Descrição do Bloco

O *charge pump* tem funcionamento ditado a partir de duas fontes de corrente, de mesmo valor em módulo, chaveadas de acordo com os sinais de entrada. A descrição em Verilog-AMS deste bloco parte da implementação destas fontes sob as condições acima. Tal comportamento é obtido através do uso de operadores condicionais *if/else* para contemplar todas as possibilidades de níveis de tensão dos sinais de entrada, desta forma, se a entrada *DOWN* for maior que um dado limiar enquanto *UP* está abaixo do mesmo, puxa-se o valor paramétrico *cur* de corrente, caso *DOWN* estiver abaixo do limiar enquanto *UP* esta acima, injeta-se este valor *cur*, caso contrário, nenhuma corrente é injetada/retirada da carga.


```

1  analog begin: main
2      @(initial_step) aux = 0.0;
3
4      enable = ((V(en) > vth) ? 0 : 1);
5
6      if (V(down)>vth && V(up)<vth)    aux = -cur;
7      else if (V(down)<vth && V(up)>vth) aux = cur;
8      else                             aux = 0;
9
10     I(aout) <+ transition(aux*enable, td, tt);
11 end

```

9.2.2 Descrição dos Pinos

Tabela 8 – Descrição dos pinos do *charge pump*.

Pino	Descrição	Tipo
<i>up</i>	Pulso de tensão fornecido pelo Q_A do PFD	<i>input</i>
<i>down</i>	Pulso de tensão fornecido pelo Q_B do PFD	<i>input</i>
<i>out</i>	Pulsos de corrente de saída do CP	<i>output</i>

9.2.3 Simulação

A simulação e os parâmetros configurados são apresentados a seguir, observa-se que foram usados os sinais de saída do PFD como entrada, a fim de gerar um cenário mais realista no contexto do PLL, e o valor *cur* igual a $20\mu\text{A}$, sendo este, um valor comumente usado em referências relacionadas a PLLs que operam na mesma faixa de frequências. A onda de saída, Fig.(49), está de acordo com o esperado, onde a corrente de saída é uma série de pulsos de magnitude igual a *cur* e período referente a diferença entre *UP* e *DOWN*, uma vez que *UP* permanece em nível alto mais tempo que *DOWN* e *DOWN* nunca sobe para nível alto enquanto *UP* está em nível baixo, validando-se assim a modelagem deste bloco.

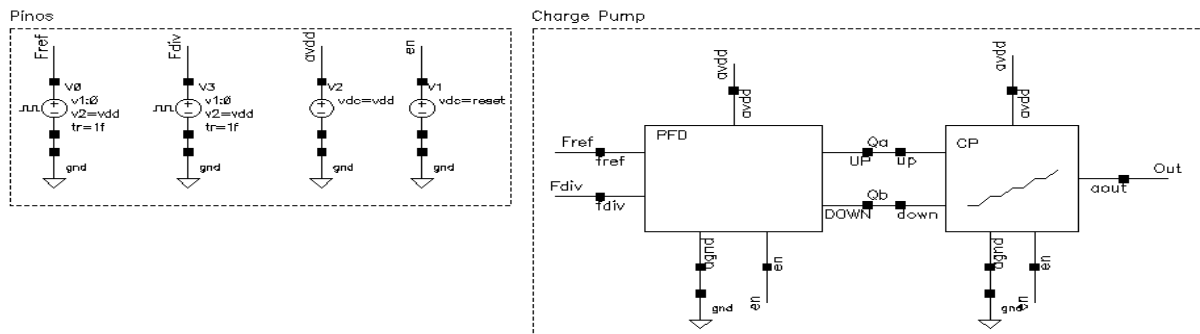
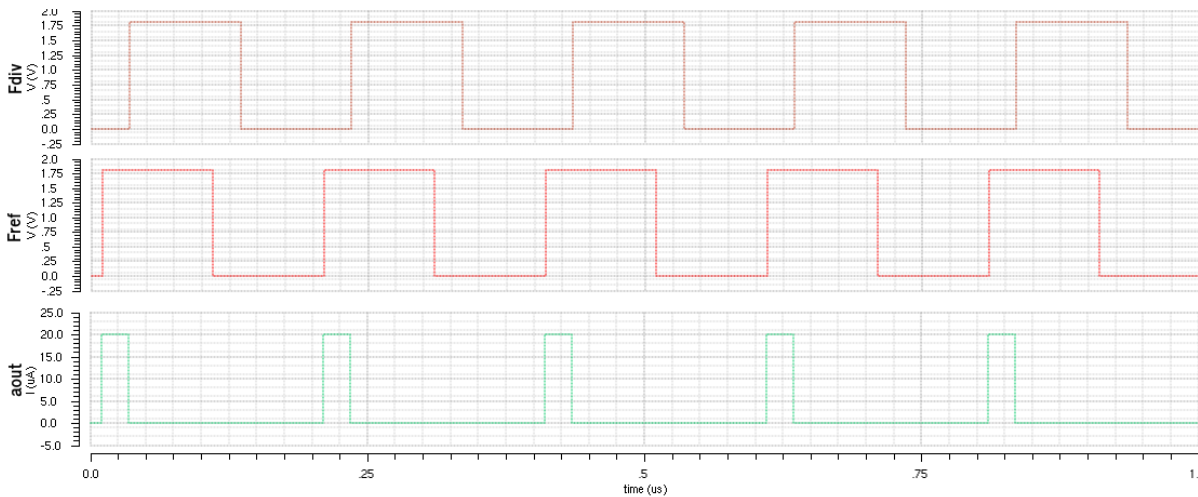


Figura 48 – Testbench do *charge pump*.

Tabela 9 – Parâmetros de simulação do *charge pump*.

Parâmetro	Descrição	Valor
corrente	Corrente de operação do <i>charge pump</i>	20 μ A
periodo	Período de operação das fontes de teste	(1/5 MHz) s
<i>delay</i>	Tempo de atraso entre as fontes de teste	25 ns
ciclos	Quantidade de ciclos usado na simulação	5

Figura 49 – Simulação do *charge pump*.

9.3 Filtro de Malha

A funcionalidade do filtro de malha está na conversão dos pulsos de corrente de sua entrada em níveis DC de tensão enquanto efetua a filtragem. A modelagem deste bloco foi feita em Verilog-AMS através de duas diretivas, conforme o código a seguir, onde foi implementada a exata disposição dos componentes passivos, e a implementação a partir da função de transferência em Laplace, porém, esta última sem sucesso. Para maiores informações do bloco e sua implementação a nível componentes físicos, [Gomes \(2015\)](#).

9.3.1 Descrição do Bloco

O filtro de malha, no contexto do PLL aqui desenvolvido, é um filtro passivo de segunda ordem, Fig.(37), sendo este responsável por gerar o nível DC de entrada do VCO. Em Verilog-AMS, a descrição deste bloco é exatamente igual à sua composição física, onde foram descritos os componentes, capacitores e resistor, individualmente, para posteriormente uní-los na mesma disposição do circuito real. Para isto, usou-se as descrições ideais de capacitores e resistores, através de suas devidas relações tensão/corrente, e o conceito de *branches* em Verilog-AMS, permitindo a união dos componentes.

```

1  analog begin: main
2      enable = ((V(en) > vth) ? 0 : 1);
3
4      V(res1) <+ r1*I(res1);
5      I(cap1) <+ c1*ddt(V(cap1));
6      I(cap2) <+ c2*ddt(V(cap2));
7  end

```

9.3.2 Descrição dos Pinos

Tabela 10 – Descrição dos pinos do filtro de malha.

Pino	Descrição	Tipo
p	Nó p do <i>branch</i> do filtro de malha	<i>inout</i>
n	Nó n do <i>branch</i> do filtro de malha	<i>inout</i>

9.3.3 Simulação

A simulação e os parâmetros configurados são apresentados a seguir, observa-se que foram usados os blocos PFD e CP como geradores do sinal de entrada, a fim de gerar condições mais realistas para o PLL. Além da simulação com o bloco descrito em HDL, fez-se também a simulação do filtro de malha com componentes ideais da biblioteca *analogLib*, obtendo-se como resultado uma onda praticamente igual a gerada anteriormente. A onda de saída, Fig.(51), está de acordo com o esperado, onde o sinal de saída é um nível DC que depende da defasagem das ondas de entrada do PFD, sendo o aspecto de serra vinculado a idealidade do modelo, validando-se assim a modelagem deste bloco.

Tabela 11 – Parâmetros de simulação do filtro de malha.

Parâmetro	Descrição	Valor
R	Valor da resistência do filtro de malha	250 K Ω
C_1	Valor do capacitor 1 do filtro de malha	16 pF
C_2	Valor do capacitor 2 do filtro de malha	1 pF
corrente	Corrente de operação do <i>charge pump</i>	20 μ A
periodo	Período de operação das fontes de teste	(1/5 MHz) s
<i>delay</i>	Tempo de atraso entre as fontes de teste	25 ns
ciclos	Quantidade de ciclos usado na simulação	10

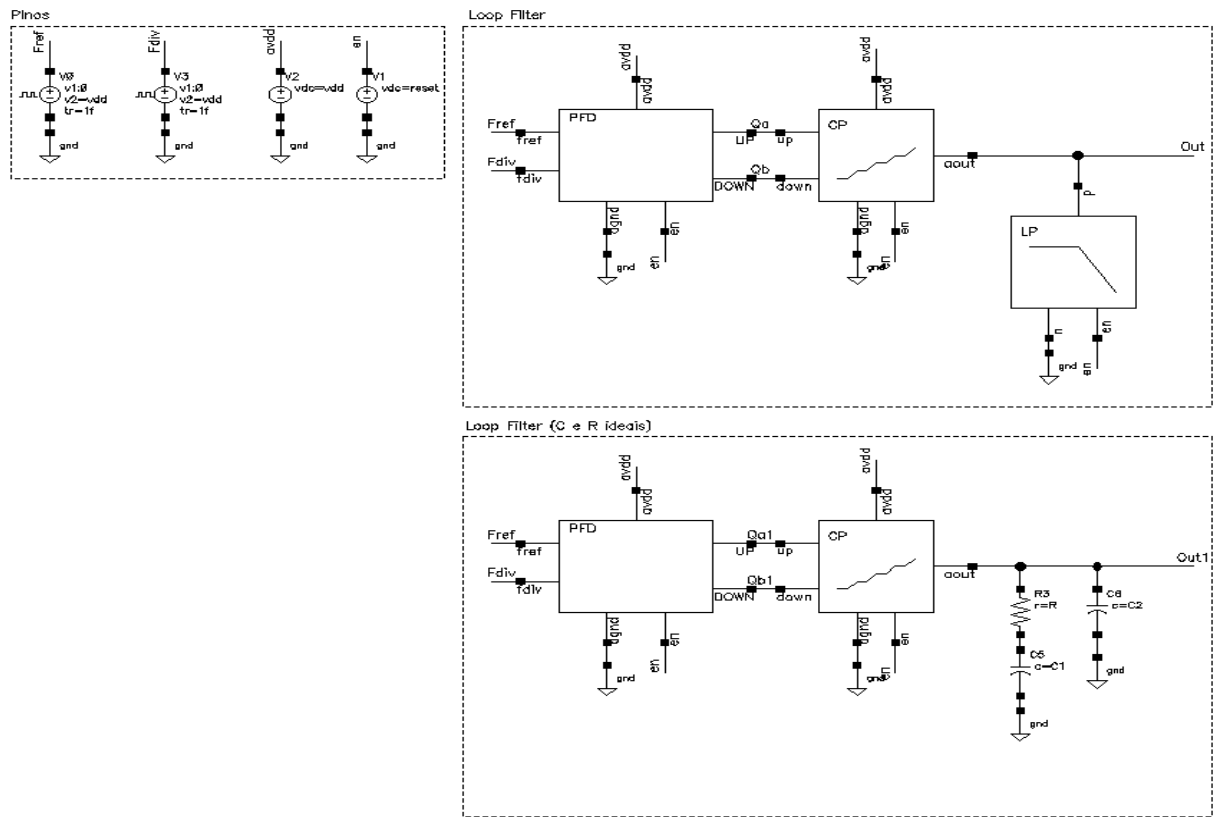


Figura 50 – Testbench do filtro de malha.

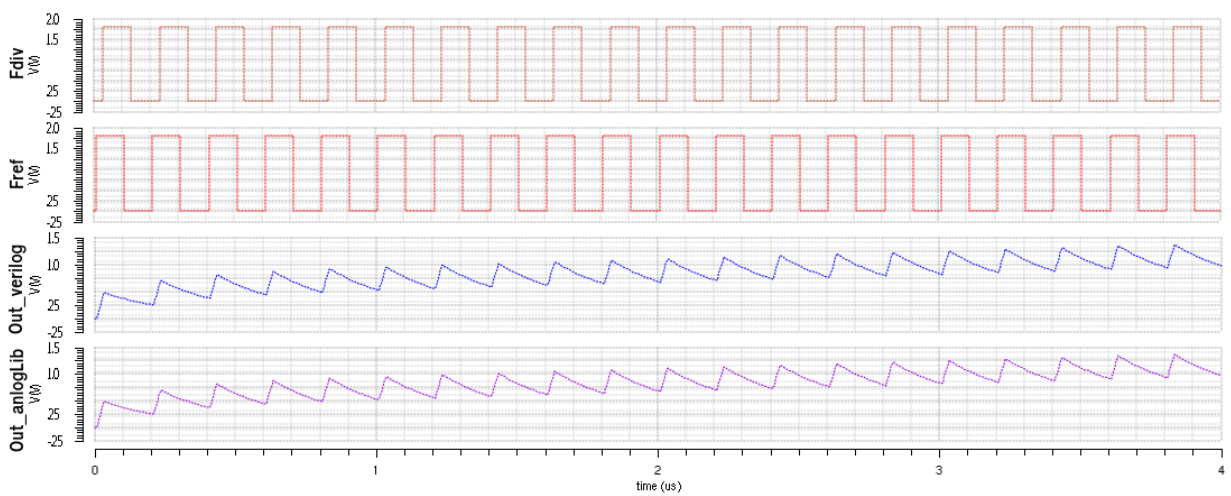


Figura 51 – Simulação do filtro de malha.

9.4 VCO

O oscilador controlado por tensão, VCO, é o bloco que gera as saídas do PLL, este converte o nível DC de entrada, gerado pelo filtro de malha, em saídas analógicas diferenciais com frequência proporcional a magnitude da tensão de entrada. A modelagem deste bloco foi feita em Verilog-AMS conforme o código apresentado na sequência. Para maiores informações do bloco e sua implementação a nível de transistores, Cap.(5).

9.4.1 Descrição do Bloco

O VCO possui funcionamento vinculado a um circuito ressonador, tanque LC, para geração de um sinal periódico, e a um circuito ativo, responsável por gerar a resistência negativa necessária para o ressonador permanecer operando. A descrição em Verilog-AMS deste bloco parte da implementação de um sistema que através de um nível DC de entrada, gera na saída um sinal periódico com fase diretamente proporcional a magnitude da tensão DC de entrada. Este comportamento é obtido através da estipulação dos parâmetros F_{min} , F_{max} , V_{min} e V_{max} , a fim de implementar uma função linear que relaciona o DC da entrada com a fase da saída através da função *idtmmod*, integração em função do tempo com operação de módulo, também referenciado como um integrador circular. Nota-se que o bloco descrito possui saídas diferenciais, ou seja, duas saídas com metade da amplitude e defasadas entre si em 180°.

```

1  analog begin: main
2      enable = (V(en) > vth) ? 0 : 1;
3
4      freq = (V(ain) - Vmin) * (Fmax - Fmin) / (Vmax - Vmin) + Fmin;
5
6      if(freq > Fmax) freq = Fmax;
7      if(freq < Fmin) freq = Fmin;
8
9      phase = 2 * M_PI * idtmmod(freq, 0.0, 1.0, -0.5);
10
11     V(aout_p) <+ (((ampl * sin(phase)) / 2) + (V(avdd) / 2)) * (enable);
12     V(aout_n) <+ ((- (ampl * sin(phase)) / 2) + (V(avdd) / 2)) * (enable);
13
14     $bound_step(0.1 / freq);
15 end

```

9.4.2 Descrição dos Pinos

Tabela 12 – Descrição dos pinos do VCO.

Pino	Descrição	Tipo
in	Nível DC de entrada de controle proveniente do filtro de malha	<i>input</i>
Out_P	Saída diferencial analógica P	<i>output</i>
Out_N	Saída diferencial analógica N	<i>output</i>

9.4.3 Simulação

As simulações, tabelas e parâmetros configurados são apresentados a seguir. Através de simulações paramétricas e das equações inseridas no ADE L, Fig.(53), Fig.(54) e Eq.(9.3) a (9.8), foi possível extrair algumas das figuras de mérito do VCO modelado, tais como: ganho (175 MHz/V), V_{ctrl} x frequência e linearidade. No entanto, os dados obtidos são ideais devido ao comportamento ideal do modelo em questão. No capítulo a seguir, será apresentada a mesma análise para o circuito real, onde será possível observar as não idealidades deste e o efeito delas no ruído de fase do bloco. No mais, o funcionamento ocorreu como esperado, validando-se assim a modelagem do mesmo.

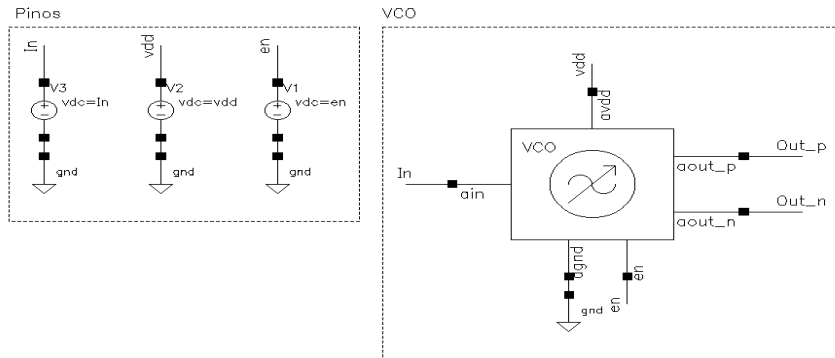


Figura 52 – Testbench do VCO.

Tabela 13 – Parâmetros de simulação do VCO.

Parâmetro	Descrição	Valor
F_{min}	Frequência máxima de saída do VCO	2.35 GHz
F_{max}	Frequência máxima de saída do VCO	2.525 GHz
Vin_dc_{min}	Nível de tensão de controle DC mínimo da entrada do VCO	0.5 V
Vin_dc_{max}	Nível de tensão de controle DC máximo da entrada do VCO	1.5 V
ampl	Amplitude de tensão dos sinais de saída do VCO	1.8 V
ciclos	Quantidade de ciclos usado na simulação	10

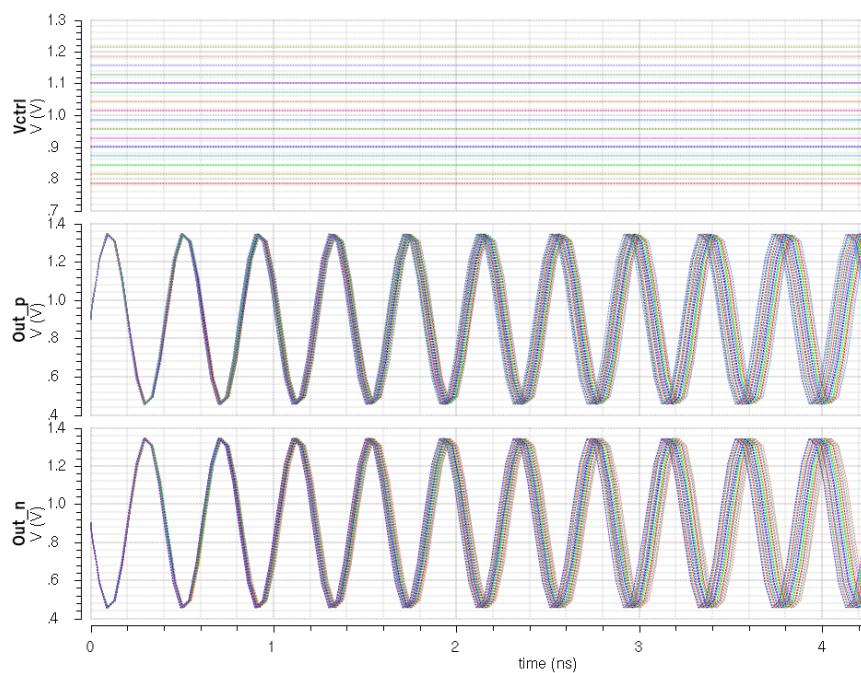


Figura 53 – Simulação do VCO para os 16 canais do PLL.

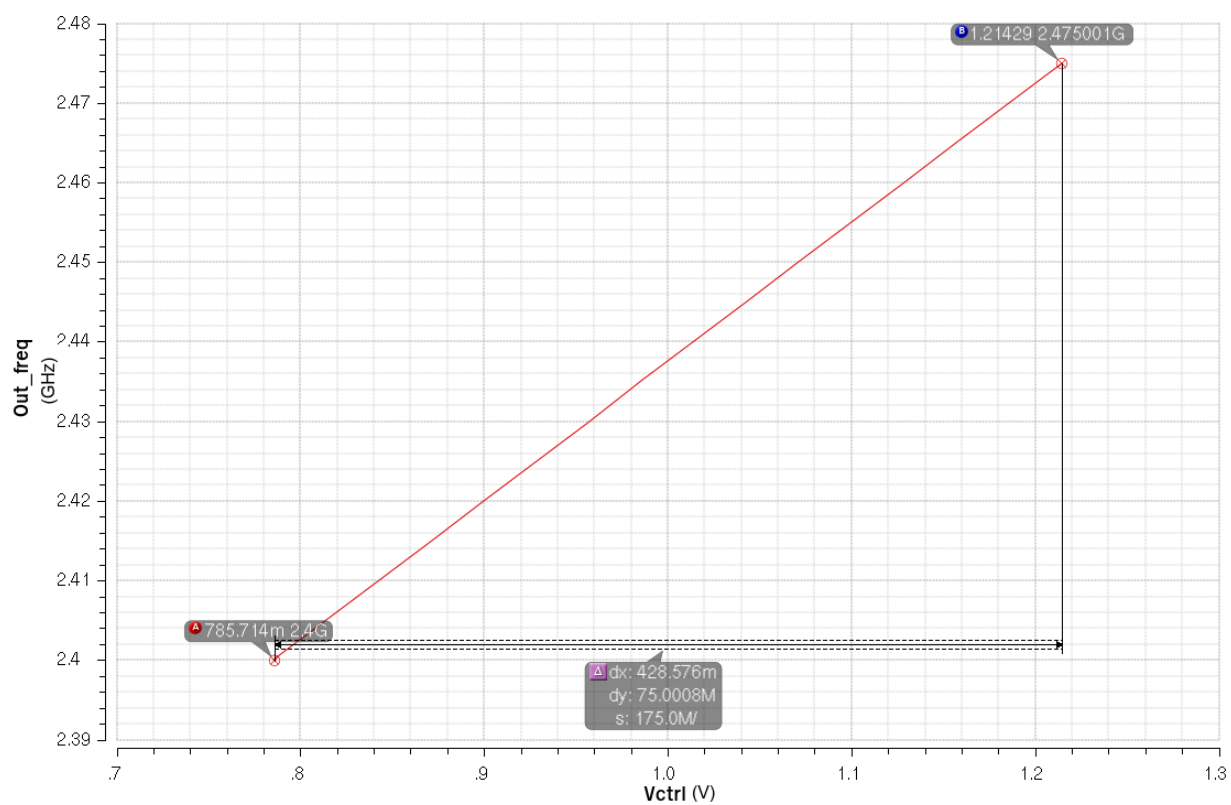


Figura 54 – Simulação do VCO - Tensão de controle X Frequência.

9.5 Conversor de Saída Diferencial para *Single Ended*

A principal função do conversor diferencial para *single ended* é justamente o que o nome diz, converter as saídas de algum circuito diferencial para uma única saída. Outra função é a conversão do sinal de analógico para digital, isto ocorre devido a um estágio de amplificação na saída do conversor que produz a saturação do sinal. A modelagem deste bloco foi feita em Verilog-AMS conforme o código apresentado na sequência. Para maiores informações do bloco e sua implementação a nível de transistores, [Pinto \(2015\)](#).

9.5.1 Descrição do Bloco

O conversor diferencial para *single ended* é um circuito de entrada diferencial e única saída cuja saída é a diferença das entradas, resultando num sinal com o dobro amplitude. A descrição em Verilog-AMS deste bloco parte da subtração do sinal de entrada p pelo n e adição de um nível DC igual a 0.9V (metade do vdd), nível DC usado em todos os blocos do PLL projetado. Posteriormente, com o sinal gerado dessa operação, utiliza-se a função *cross* para realizar a saturação do sinal, de forma que a partir do instante que o sinal passa do limiar ($0.5 \cdot vdd$), este equivale a vdd, e para valores menores a esse limiar, o sinal passa a ser igual a gnd.

```

1  analog begin: main
2      enable = ((V(en) > vth) ? 0 : 1);
3
4      vout = ((V(ain_p) - V(ain_n)) + V(avdd)/2);
5
6      @(cross(vout - vth));
7      voutd = ((vout > vth) ? V(avdd) : V(agnd));
8
9      V(dout) <+ transition(voutd*(enable), td, tt);
10 end

```

9.5.2 Descrição dos Pinos

Tabela 14 – Descrição dos pinos do conversor de saída diferencial para *single ended*.

Pino	Descrição	Tipo
ain_p	Entrada diferencial analógica P	<i>input</i>
ain_n	Entrada diferencial analógica N	<i>input</i>
dout	Saída <i>single ended</i> digital	<i>output</i>

9.5.3 Simulação

A simulação e os parâmetros configurados para tal estão apresentados a seguir, observa-se que foram usados sinais configurados similarmente àqueles que foram gerados pelas saídas do VCO. A onda de saída, Fig.(56), esta de acordo com o esperado, onde a onda de saída está saturada e com o dobro da amplitude das ondas de entrada, validando-se assim a modelagem deste bloco.

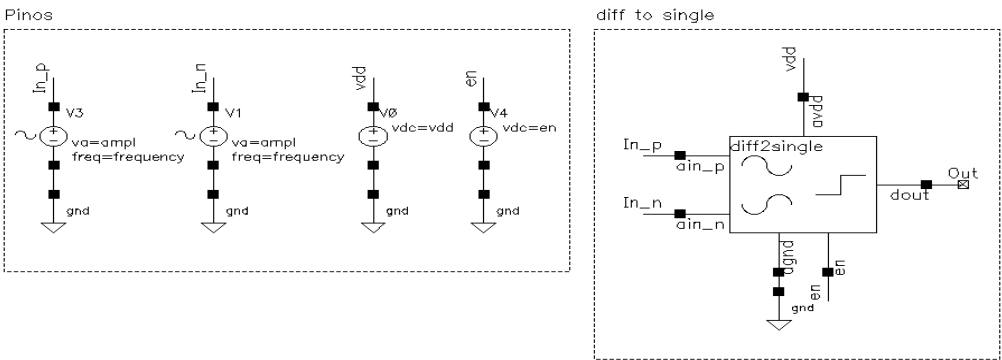


Figura 55 – Testbench do conversor de saída diferencial para *single ended*.

Tabela 15 – Parâmetros de simulação do conversor de saída diferencial para *single ended*.

Parâmetro	Descrição	Valor
frequência	Frequência de operação da fonte de teste	4.8G V
nivel_dc	Nível DC de operação da fonte de teste	0.9 V
ampl	Amplitude de tensão dos sinais de saída do VCO	0.45 V
ciclos	Quantidade de ciclos usado na simulação	5

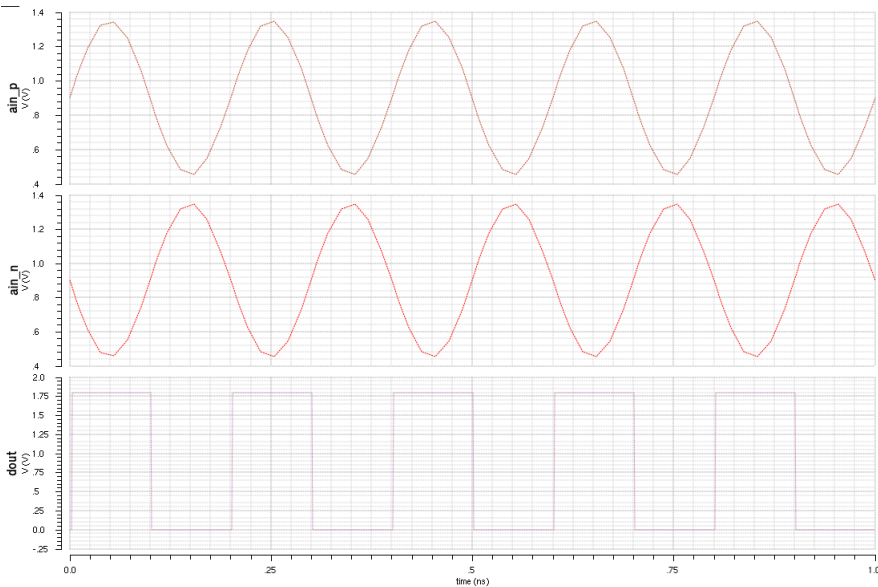


Figura 56 – Simulação do conversor de saída diferencial para *single ended*.

9.6 Divisor de Frequência

O bloco divisor de frequências é encarregado de realizar a divisão de frequência de um sinal genérico de entrada. No caso do PLL, o divisor está localizado na realimentação e divide a frequência do sinal de saída do PLL para que este possa ser comparado com a referência pelo PFD. A modelagem deste bloco foi feita a partir da descrição em Verilog-AMS dos componentes internos, como portas lógicas, multiplexadores e *flip flops*, e posterior junção deles para o funcionamento do sistema completo. Para maiores informações do divisor, componentes internos e sua implementação a nível de transistores, [A.1](#) e [Pinto \(2015\)](#).

9.6.1 Descrição do Bloco

O divisor de frequência é composto por uma série de contadores e portas lógicas que a partir de uma palavra binária de n bits, seleciona um determinado fator de divisão correspondente a um dos 2^n canais sintonizáveis. Na aplicação do PLL para transceptor ZigBee, a palavra de seleção possui 4 bits e permite a sintonização de 16 canais, espaçados entre si em 5 MHz, com frequência de operação na faixa de 2.4 GHz a 2.475 GHz.

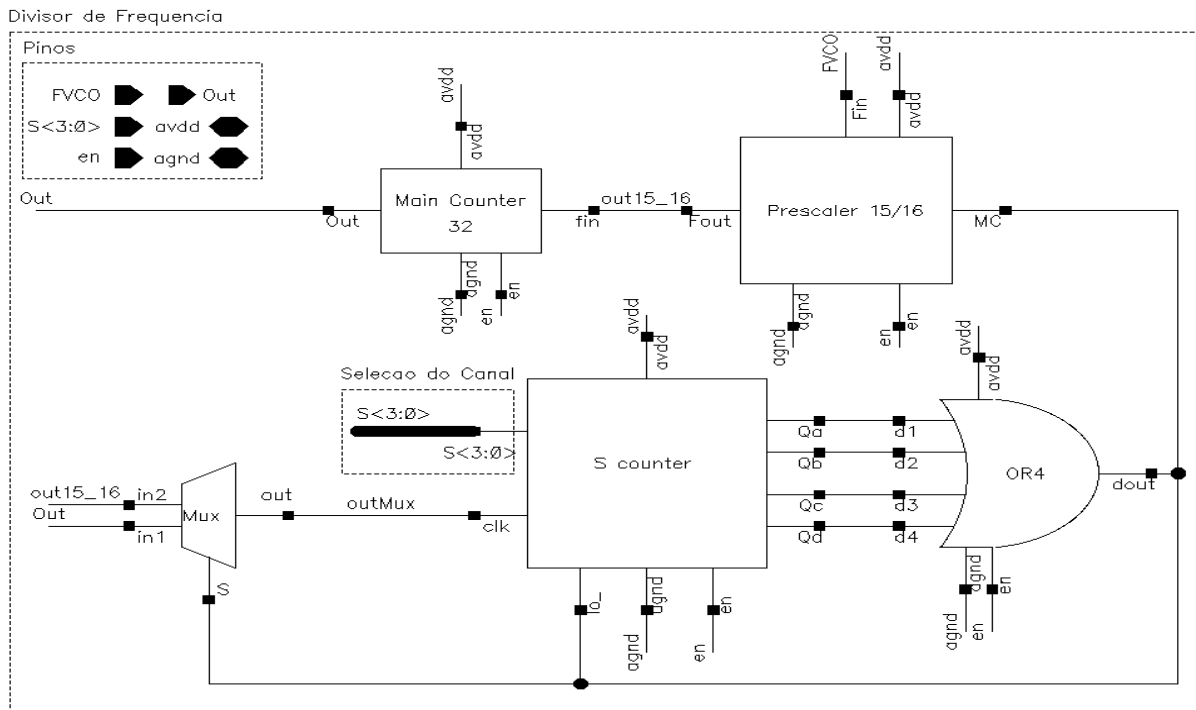


Figura 57 – Esquemático do Divisor de Frequência.

9.6.2 Descrição dos Pinos

Tabela 16 – Descrição dos pinos do Divisor de Frequência.

Pino	Descrição	Tipo
F_{VCO}	Entrada digital proveniente do VCO com metade da frequência	<i>input</i>
$S < 4 >$	Vetor de entrada digital para seleção do canal de saída do PLL	<i>input</i>
out	Saída digital com frequência relativa ao canal escolhido	<i>output</i>

9.6.3 Simulação

A simulação e os parâmetros configurados são apresentados na sequência, observa-se que o sinal de entrada da simulação possui frequência próxima ao esperado para o canal 11 (0000) do ZigBee, onde a partir desse sinal, plotou-se no mesmo gráfico os 16 fatores de divisão possíveis. A onda de saída, Fig.(59), esta de acordo com o esperado, pois com o acréscimo do fator de divisão o período da onda de saída torna-se menor. A validação dos valores de frequências e fatores de divisão foi feita a partir das Eq.(9.3) a (9.8) no ADE L, como pode ser visto na simulação do PLL completo, encontrado na seção (9.7).

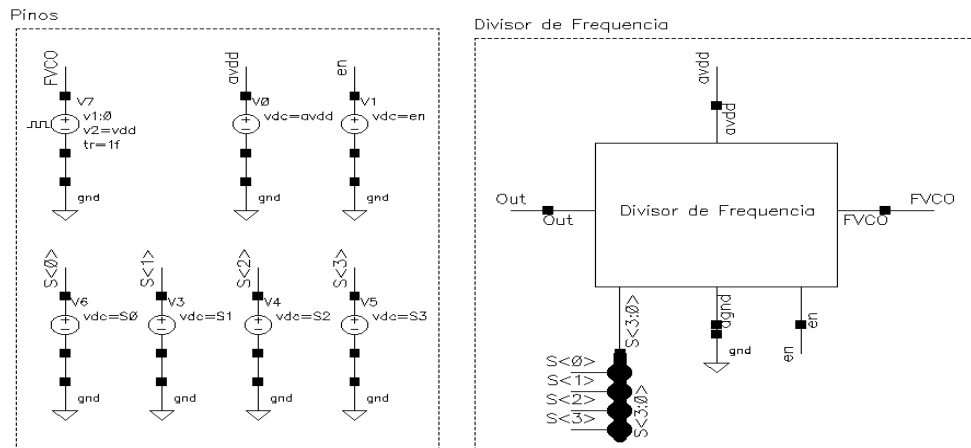


Figura 58 – Testbench do Divisor de Frequência.

Tabela 17 – Parâmetros de simulação do Divisor de Frequência.

Parâmetro	Descrição	Valor
S0	Fonte relativa ao bit 0 da seleção do canal	0/1.8 V
S1	Fonte relativa ao bit 1 da seleção do canal	0/1.8 V
S2	Fonte relativa ao bit 2 da seleção do canal	0/1.8 V
S3	Fonte relativa ao bit 3 da seleção do canal	0/1.8 V
periodo	Período de operação da fonte de teste	(1/2.4 GHz) s

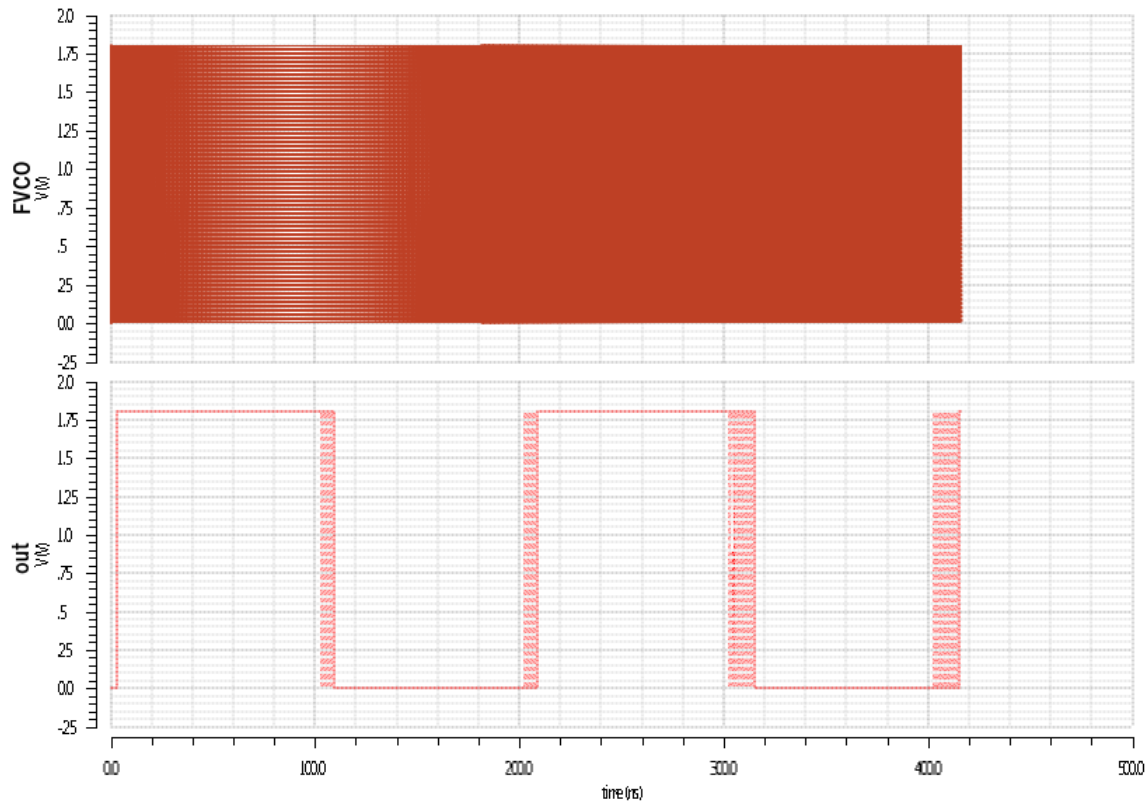


Figura 59 – Simulação do Divisor de Frequência para os 16 canais.

9.7 PLL completo

A modelagem do PLL completo foi feita a partir da união dos blocos modelados nas seções anteriores. Na sequência, serão apresentados os dados extraídos através das simulações realizadas com os blocos modelados em Verilog-AMS. Posteriormente, serão mostrados os resultados das simulações mistas, no qual blocos reais, a nível de transistores, foram simulados juntamente com os blocos em Verilog-AMS.

9.7.1 Descrição do Bloco

O funcionamento detalhado do PLL já foi descrito anteriormente, no entanto, vale relembrar que o objetivo é realizar a sintonização em frequência a partir de uma fonte de sinal de referência e uma chave de seleção do canal. Isto ocorre através da correção contínua da diferença de fase e/ou frequência existente entre o sinal de referência e o sinal de saída do divisor. Para a aplicação em questão, tem-se um sinal referência de 5 MHz e uma palavra binária de 4 bits para efetuar a sintonização de 16 canais, distribuídos na banda de frequência localizada entre 2.4 a 2.475 GHz.

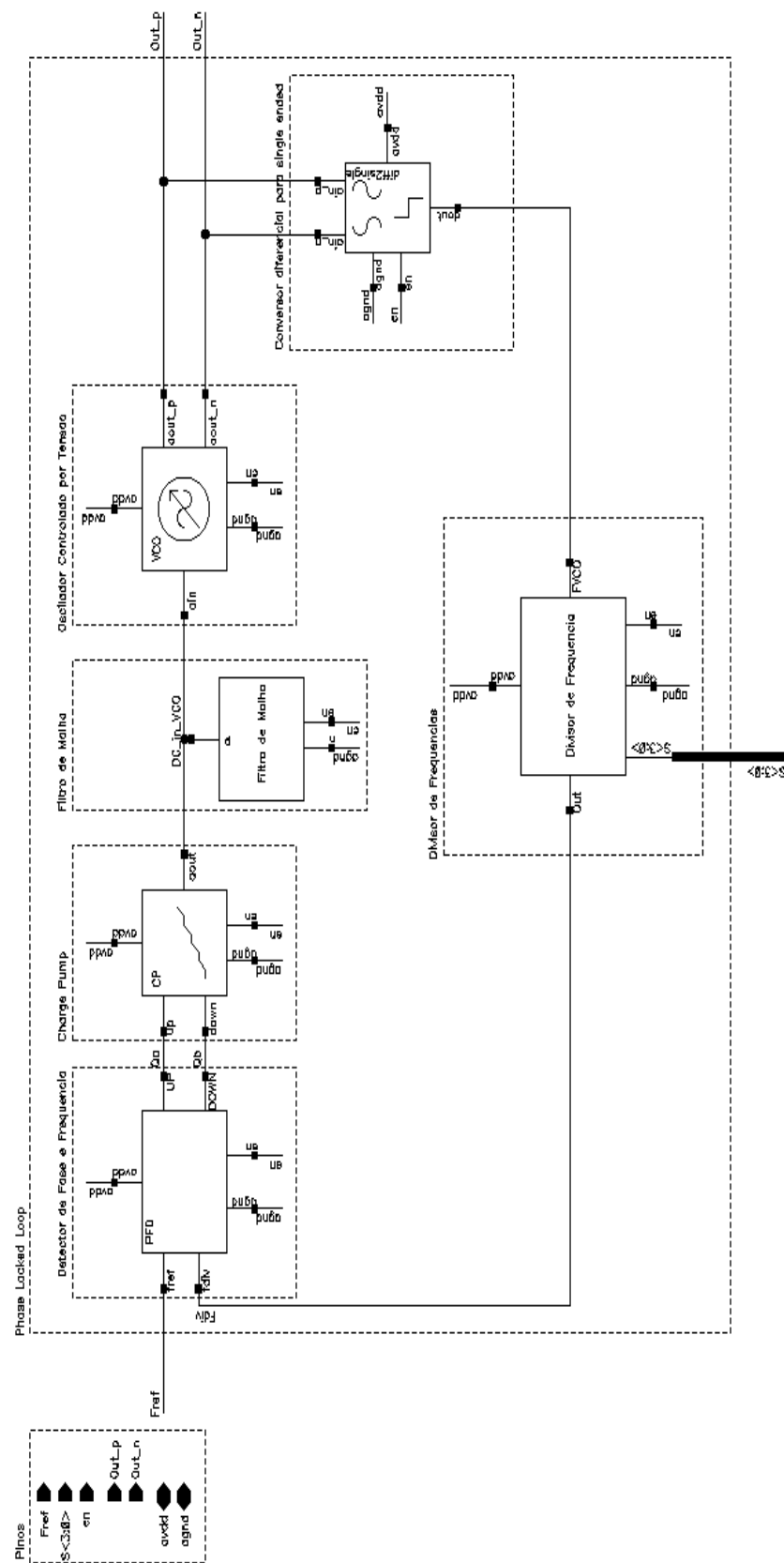


Figura 60 – Esquemático do PLL.

9.7.2 Descrição dos Pinos

Tabela 18 – Descrição dos pinos do PLL completo.

Pino	Descrição	Tipo
F_{ref}	Entrada digital proveniente do sinal de referência	<i>input</i>
$S < 4 >$	Vetor de entrada digital para seleção do canal de saída	<i>input</i>
Out_P	Saída diferencial analógica P	<i>output</i>
Out_N	Saída diferencial analógica N	<i>output</i>

9.7.3 Simulação

Na sequência estão dispostas tabelas, equações, simulações e estados de simulação gerados com base no modelo completo do PLL. Vale ressaltar que foi realizada a simulação de cada um 16 canais, onde foi obtido sucesso para todos, ou seja, o sistema alcançou a estabilidade com base nos valores achados no procedimento de projeto do PLL Cap.(7).

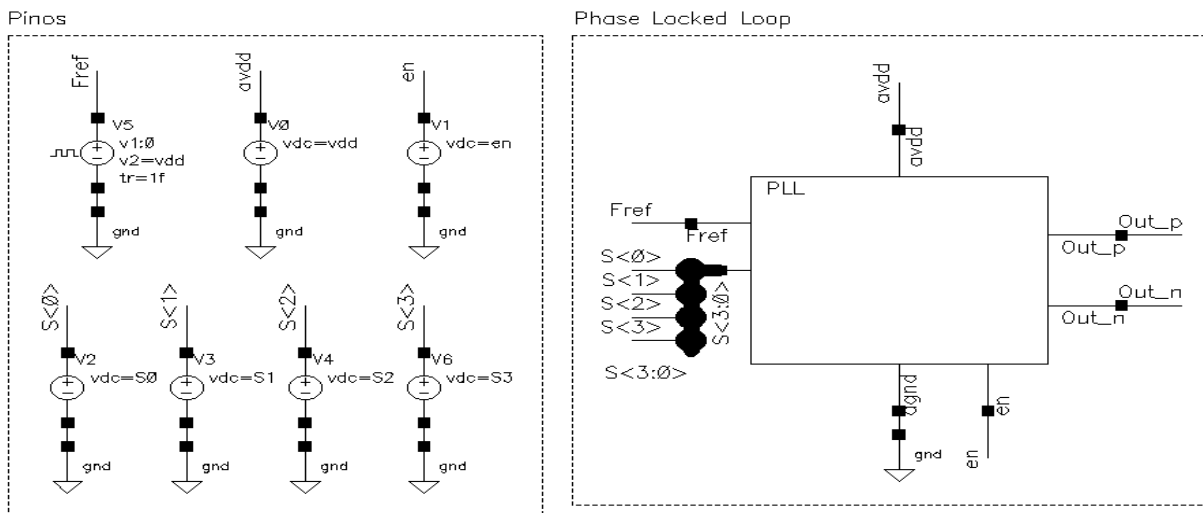


Figura 61 – Testbench do PLL modelado.

As formas de onda e os estados de simulação do ADE L a seguir são referentes aos canais 11, 18 e 26, respectivamente, ilustrando o funcionamento do modelo para os canais extremos e um dos canais do meio. Na simulação, encontra-se as ondas F_{ref} , F_{div} , VCO_{in} e as saídas diferenciais do PLL, Out_P e Out_N . O objetivo é que com o passar dos ciclos de simulação a onda F_{div} se aproxime a F_{ref} , o que significa que o sistema entra em estado de *lock*, travando a frequência desejada na saída. Observa-se que a onda tocante ao nível DC de entrada do VCO, VCO_{in} , é um indicador de estabilidade e do *settling time* do PLL, sendo esta um importante parâmetro de análise do sistema, de forma que seu formato também indica a ordem do filtro, Fig.(37), no caso segunda ordem, e o fator de amortecimento usado no projeto, igual a 1.



Figura 62 – Simulação da modelagem do PLL para o canal 11 ($S < 4 \Rightarrow 0000$).

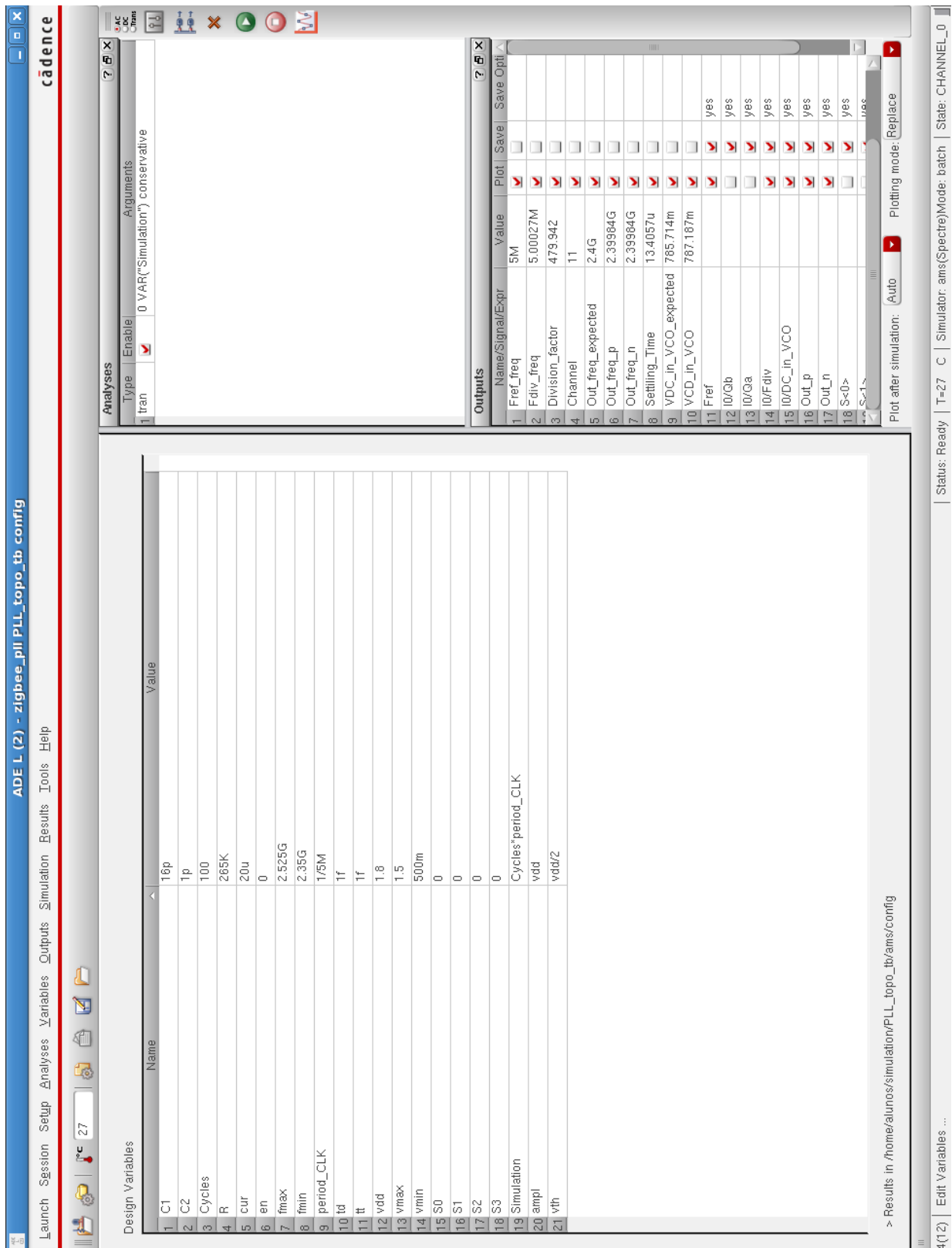


Figura 63 – ADE L para simulação da modelagem do PLL para o canal 11 ($S < 4 \Rightarrow 0000$).



Figura 64 – Simulação da modelagem do PLL para o canal 18 ($S < 4 \Rightarrow 0111$).



Figura 65 – ADE L para simulação da modelagem do PLL para o canal 18 ($S < 4 \geq 0111$).



Figura 66 – Simulação da modelagem do PLL para o canal 26 ($S < 4 \Rightarrow 1111$).



Figura 67 – ADE L para simulação da modelagem do PLL para o canal 26 ($S < 4 \Rightarrow 1111$).

Tabela 19 – Parâmetros de simulação do PLL completo.

Parâmetro	Descrição	Valor
S0	Fonte relativa ao bit 0 da seleção do canal	0/1.8 V
S1	Fonte relativa ao bit 1 da seleção do canal	0/1.8 V
S2	Fonte relativa ao bit 2 da seleção do canal	0/1.8 V
S3	Fonte relativa ao bit 3 da seleção do canal	0/1.8 V
periodo_ F_{ref}	Período de operação das fontes de teste	(1/5 MHz) s
ciclos	Quantidade de ciclos de simulação em relação a 2.4 GHz	100 s

A Fig.(68) se refere as saídas do PLL para o canal 18, atenta-se em sua forma senoidal e na defasagem de 180° entre elas, caracterizando um circuito com saídas diferenciais analógicas. Para este canal temos como frequência esperada:

$$F_{CANAL} = 2400 + 5(18 - 11)MHz = 2.435 GHz. \quad (9.1)$$

De maneira que na Fig.(68) temos:

$$F_{CANAL} = \frac{1}{dx} = \frac{1}{410.9066648 \cdot 10^{-12}} \approx 2.434 GHz, \quad (9.2)$$

onde este valor é dependente do período escolhido, pois a cada ciclo o sistema tende a se aproximar mais do valor esperado.

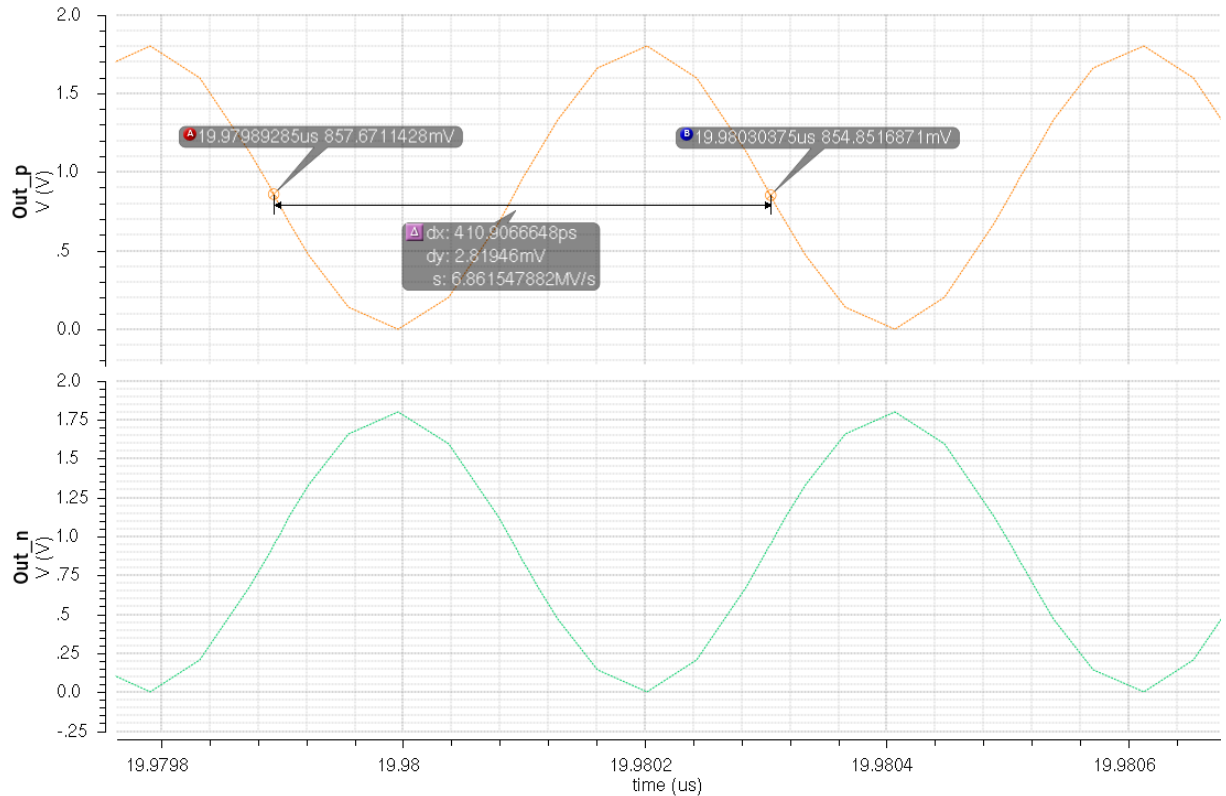


Figura 68 – Sinais de saída da simulação da modelagem do PLL para o canal 18 ($S < 4 \geq 0111$).

9.7.3.1 Equações

Usou-se expressões junto ao simulador ADE L, tais expressões estão apresentadas como as “pseudo” fórmulas 9.3 a 9.8, sendo estas uma alusão às funções usadas na calculadora do Cadence para geração dos dados a partir das ondas de saída.

$$\begin{aligned} Canal = & \left[y_{max}(S[0]) \cdot \left(\frac{1}{vdd} \right) \right] \cdot 2^0 + \left[y_{max}(S[1]) \cdot \left(\frac{1}{vdd} \right) \right] \cdot 2^1 + \left[y_{max}(S[2]) \cdot \left(\frac{1}{vdd} \right) \right] \cdot 2^2 \\ & + \left[y_{max}(S[3]) \cdot \left(\frac{1}{vdd} \right) \right] \cdot 2^3 + 11 \end{aligned} \quad (9.3)$$

$$Frequencia = \frac{1}{\left[cross(sinal, \frac{vdd}{2}, 2, rising) - cross(sinal, \frac{vdd}{2}, 1, rising) \right]} \quad (9.4)$$

$$Out_freq_esperada = 2.4 \text{ GHz} + [(11 + Canal) \cdot 5 \text{ MHz}] \quad (9.5)$$

$$\begin{aligned} Fator_De_Divisao = & \left[cross \left(F_{div}, \frac{vdd}{2}, 2, rising \right) - cross \left(F_{div}, \frac{vdd}{2}, 1, rising \right) \right] \\ & - \left[cross \left(Out, \frac{vdd}{2}, 2, rising \right) - cross \left(Out, \frac{vdd}{2}, 1, rising \right) \right] \end{aligned} \quad (9.6)$$

$$Tensao_DC_VCO_esperada = \frac{(Out_freq_esperada - 2.2625 \text{ GHz})}{0.175} \quad (9.7)$$

$$Tensao_DC_VCO = average(DC_in_VCO(clip(15\mu s, 20\mu s))) \quad (9.8)$$

9.7.3.2 Tabelas

As tabelas 20, 21, 22 e 23 mostram os resultados obtidos quanto a frequência de saída, fator de divisão, tensão DC na entrada do VCO e *settling time*, respectivamente. Nota-se que os valores obtidos são muito próximos aos esperados, o que valida a modelagem do sistema.

Tabela 20 – Frequência de Saída do PLL modelado.

Canal	Frequência de Saída Esperada	Frequência de Saída Obtida
11	2400 MHz	2.39984 GHz
12	2405 MHz	2.40494 GHz
13	2410 MHz	2.41006 GHz
14	2415 MHz	2.41521 GHz
15	2420 MHz	2.42040 GHz
16	2425 MHz	2.42483 GHz
17	2430 MHz	2.42995 GHz
18	2435 MHz	2.43509 GHz
19	2440 MHz	2.44027 GHz
20	2445 MHz	2.44549 GHz
21	2450 MHz	2.44980 GHz
22	2455 MHz	2.45484 GHz
23	2460 MHz	2.46011 GHz
24	2465 MHz	2.46532 GHz
25	2470 MHz	2.47057 GHz
26	2475 MHz	2.47477 GHz

Tabela 21 – Fator de Divisão do PLL modelado.

Canal	Fator de Divisão Esperado	Fator de Divisão Obtido
11	480	479.942
12	481	480.961
13	482	481.985
14	483	483.014
15	484	484.050
16	485	484.935
17	486	485.958
18	487	486.987
19	488	488.022
20	489	489.065
21	490	489.926
22	491	490.953
23	492	491.987
24	493	493.027
25	494	494.076
26	495	494.915

Tabela 22 – Tensão de Controle do VCO.

Canal	Tensão de Controle Esperada	Tensão de Controle Obtida
11	785.714 mV	787.187 mV
12	814.286 mV	815.801 mV
13	842.857 mV	844.416 mV
14	871.429 mV	873.033 mV
15	900.000 mV	901.651 mV
16	928.571 mV	930.270 mV
17	957.143 mV	958.889 mV
18	985.714 mV	987.509 mV
19	1.01429 V	1.01613 V
20	1.04286 V	1.04475 V
21	1.07143 V	1.07337 V
22	1.10000 V	1.10199 V
23	1.12857 V	1.13061 V
24	1.15714 V	1.15922 V
25	1.18571 V	1.18783 V
26	1.21429 V	1.21643 V

Tabela 23 – *Settling Time* do PLL modelado.

Canal	Palavra de Seleção (S)	<i>Settling Time</i>
11	0000	13.4057 μ s
12	0001	13.4049 μ s
13	0010	13.4041 μ s
14	0011	13.4025 μ s
15	0100	13.4030 μ s
16	0101	13.4026 μ s
17	0110	13.4022 μ s
18	0111	13.4018 μ s
19	1000	13.4015 μ s
20	1001	13.2069 μ s
21	1010	13.2066 μ s
22	1011	13.2063 μ s
23	1100	13.2059 μ s
24	1101	13.2054 μ s
25	1110	13.2045 μ s
26	1111	13.2032 μ s

10 Simulações Mistas

Num contexto mais amplo, seguindo a metodologia de projeto *Top-Down*, o projeto do PLL consiste na modelagem do mesmo em Verilog-AMS e no projeto de todos os blocos que contemplam o sistema a nível de transistores. Para tal, existem 3 pessoas trabalhando no projeto, de forma que a modelagem do PLL e projeto do VCO foram desenvolvidos neste trabalho, os projetos elétricos do PFD, *charge pump* e filtro de malha foram realizados em [Gomes \(2015\)](#) e o Divisor de Frequências e conversor diferencial para *single ended* em [Pinto \(2015\)](#).

A partir deste ponto, onde a modelagem está concluída, é possível realizar simulações no qual uma parte dos blocos é representada em nível de transistor e o restante utilizando Verilog-AMS, com a finalidade de analisar o comportamento no âmbito do sistema completo. Infelizmente, por diversos fatores não foi possível simular todos os blocos. No entanto, os blocos PFD, *charge pump*, filtro de malha e conversor diferencial para *single ended* foram simulados a nível de sistema através deste tipo de simulação, os resultados são apresentados a seguir.

10.1 PFD, *Charge Pump* e Filtro de Malha

As condições de simulação são as mesmas da Fig.(64), com a modificação que os blocos PFD, *charge pump* e filtro de malha foram substituídos por seus esquemáticos elétricos. Observa-se que o circuito estabilizou, a frequência de saída está muito próxima à esperada e a onda referente a tensão DC da entrada do VCO teve uma certa modificação, onde o valor do *settling time* está menor. Vale ressaltar que a estimativa de consumo de potência destes 3 blocos juntos é de 91.6309 μW .

Outputs		
	Name/Signal/Expr	Value
1	Fref_freq	5M
2	Fdiv_freq	5.00014M
3	Division_factor	486.998
4	Channel	18
5	Out_freq_expected	2.435G
6	Out_freq_p	2.43506G
7	Out_freq_n	2.43506G
8	Settling_Time	10.4012u
9	VDC_in_VCO_expected	985.714m
10	VCD_in_VCO	986.189m

Figura 69 – Resultados das equações no ADE L da simulação abaixo.

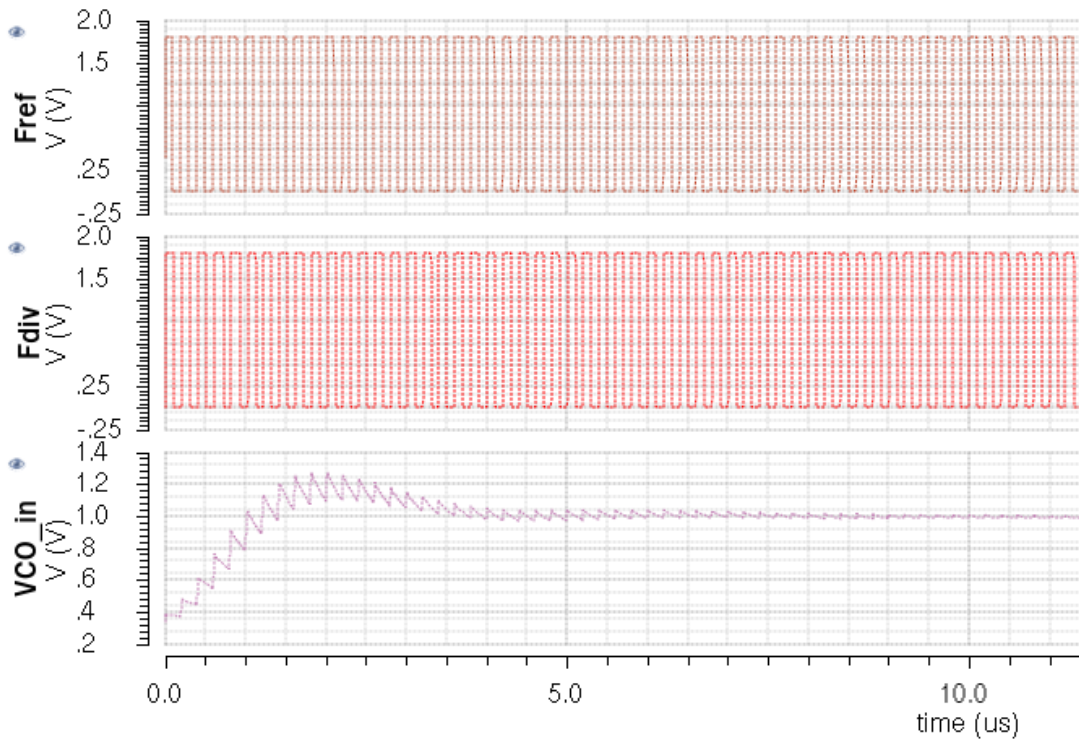


Figura 70 – Simulação mista do PFD, *charge pump* e filtro de malha com o modelo do PLL completo para o canal 18 ($S < 4 \Rightarrow 0111$).

10.2 Conversor Diferencial para *Single Ended*

Novamente, as condições de simulação são as mesmas da Fig.(64), com a modificação que o bloco conversor diferencial para *single ended* agora é real, a nível de componentes elétricos. Na simulação, Fig.(71), é possível observar que o circuito não chegou a estabilidade, apesar de estar tendendo a ela, assim como os parâmetros do ADE L, impossibilitando o cálculo do *settling time*. Vale ressaltar que a estimativa de consumo de potência deste bloco, para a maior frequência estipulada no modelo do PLL, 2.525 GHz, é de 874.188 μ W.

Outputs		
	Name/Signal/Expr	Value
1	Fref_freq	5M
2	Fdiv_freq	5.01089M
3	Division_factor	486.76
4	Channel	18
5	Out_freq_expected	2.435G
6	Out_freq_p	2.4391G
7	Out_freq_n	2.43981G
8	VDC_in_VCO_expected	985.714m
9	VCD_in_VCO	995.606m

Figura 71 – Resultados das equações no ADE L da simulação abaixo.

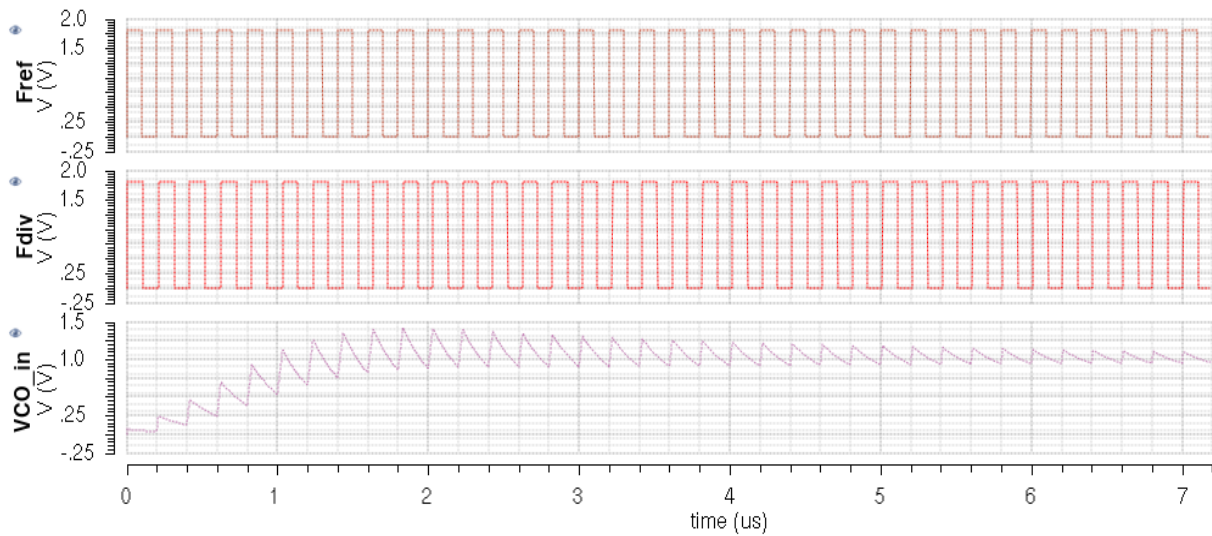


Figura 72 – Simulação mista do conversor diferencial para *single ended* com o modelo do PLL completo para o canal 18 ($S < 4 \Rightarrow 0111$).

10.3 Divisor de Frequências e Conversor Diferencial para *Single Ended*

Para o divisor, não foi possível realizar a simulação em conjunto com o modelo do PLL, devido a limitação de memória do computador usado. Portanto, a simulação feita tem como objetivo analisar seu comportamento em conjunto com o conversor diferencial para *single ended* e explicitar as diferenças entre o modelo em Verilog-AMS e o circuito a nível de transistores. Os resultados obtidos foram muito bons, onde o circuito projetado possui um erro muito pequeno, aproximando-se bem do comportamento do modelo em alto nível, o que tende a deixar uma boa impressão quanto ao funcionamento do mesmo no contexto do PLL completo. Vale ressaltar que a estimativa de consumo de potência deste bloco para o pior caso, canal 26, é de 2.18871 mW.

Outputs	
Name/Signal/Expr	Value
1 FVCO	
2 Out	
3 Out1	
4 Freq_FVCO	2.52502G
5 Fator_Divisao_Real	487.005
6 Fator_Divisao_Verilog	487.005
7 Freq_Out_Real	5.1848M
8 Freq_Out_Verilog	5.1848M

Figura 73 – Resultados das equações no ADE L da simulação abaixo.

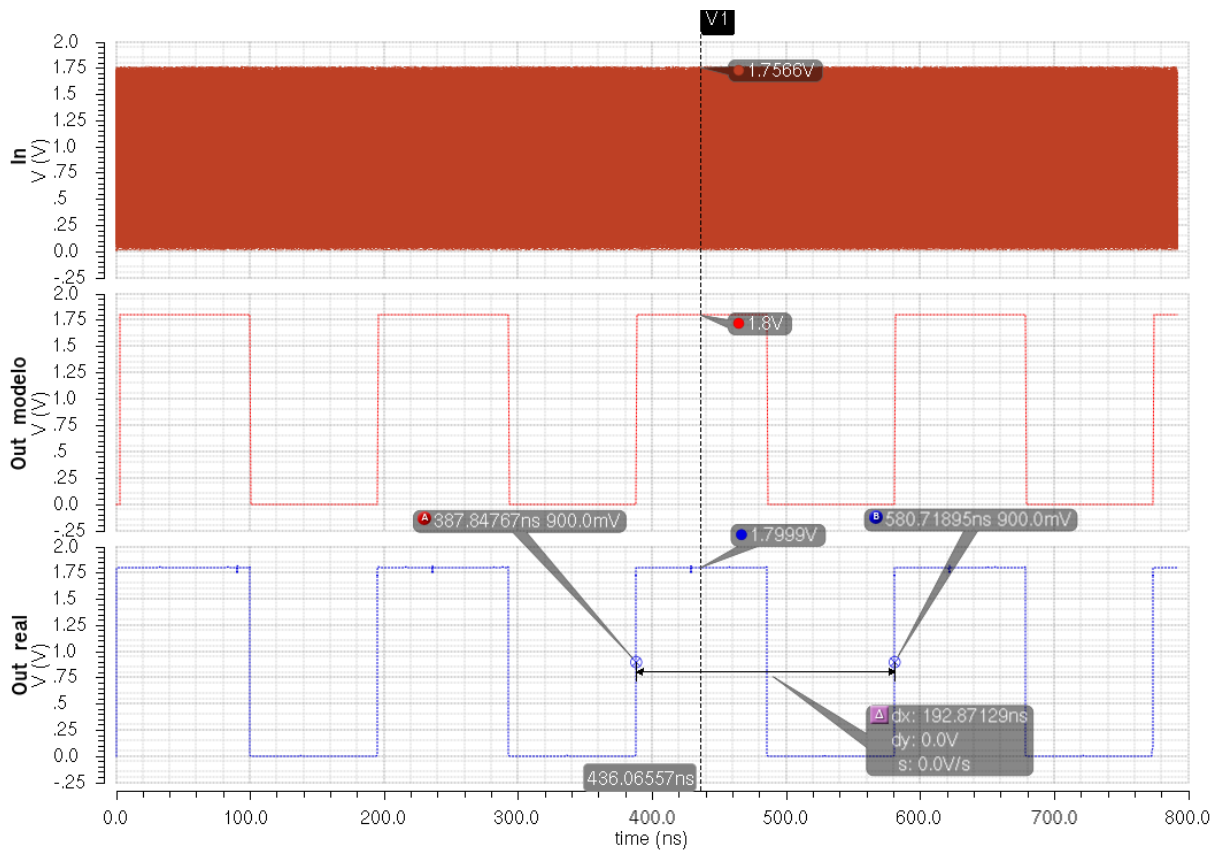


Figura 74 – Simulação mista entre os divisores de frequência modelado e projetado.

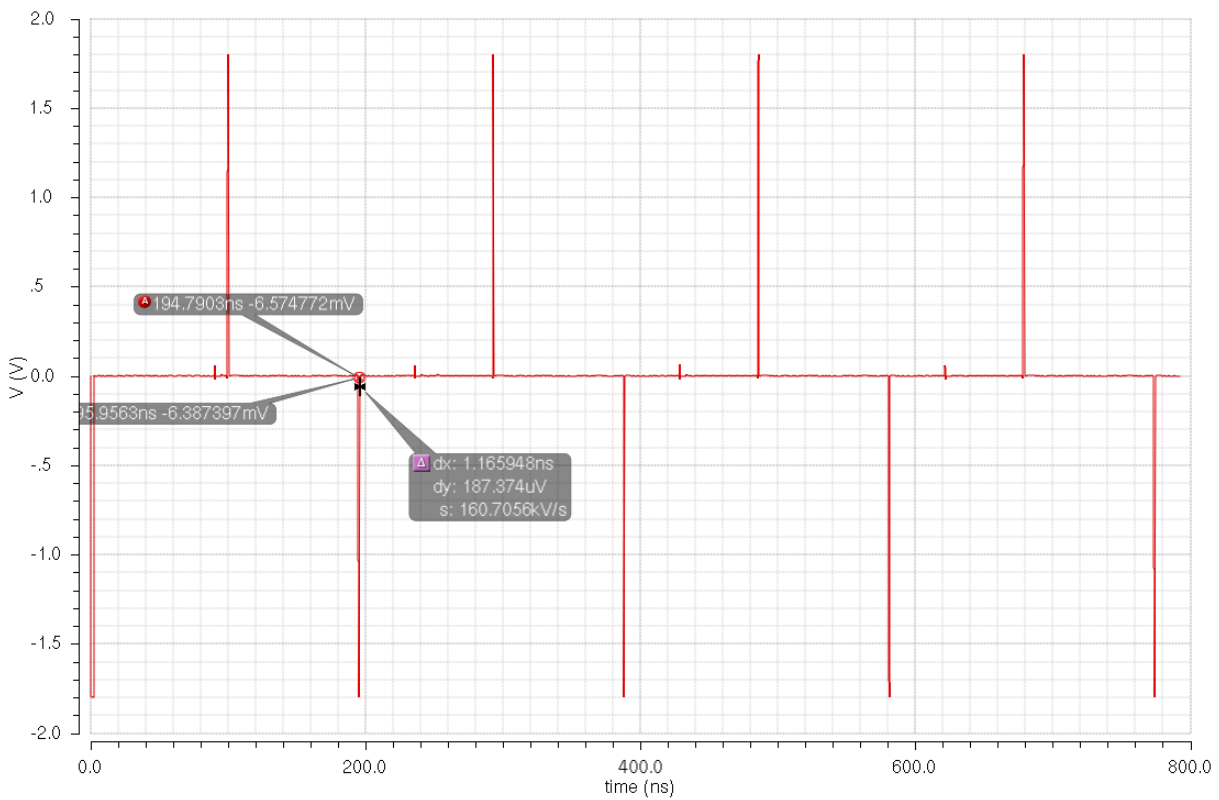


Figura 75 – Erro entre os divisores de frequência modelado e projetado.

Parte III

Conclusão

11 Conclusão

Em função da demanda de sistemas de comunicação sem fio que integrem baixo alcance, custo, confiabilidade e segurança, o protocolo ZigBee aparece como uma boa alternativa, implementando redes de sensoriamento sem fio (WSN). O ZigBee, especificado pela norma IEEE 802.15.4, utiliza a banda de rádio ISM, em âmbito global, atuando em 2.4GHz. Desta forma, este trabalho tem como objetivo principal, a modelagem de um PLL e projeto de um VCO para um transceptor ZigBee, através do uso do fluxo de projeto das ferramentas profissionais Cadence e da metodologia de projeto *Top-Down*.

Ao longo da primeira parte deste manuscrito, apresentaram-se pontos chave para elaboração do trabalho. Abordou-se informações sobre o protocolo ZigBee; funcionamento, características, tipos, metodologia de projeto e figuras de mérito do PLL; características principais, funcionamento, topologias e ruído de fase do VCO; fundamentos da metodologia de projeto *Top-Down*; e conceitos básicos da linguagem Verilog-AMS. A partir da fundamentação teórica e baseando-se na aplicação do protocolo ZigBee, fez-se a proposta do tipo de PLL que foi modelado e das possíveis topologias de VCO condizentes com as especificações. Por fim, de acordo com várias referências, formou-se uma metodologia de projeto para o PLL escolhido e levantou-se alguns parâmetros preliminares do projeto.

Na segunda parte, primeiramente foi feito o planejamento da modelagem, definindo aspectos mais detalhados dos blocos. Posteriormente, efetuou-se a modelagem em Verilog-AMS dos mesmos, de maneira que cada um foi validado através de simulações condizentes ao âmbito da aplicação. Em sequência, projetou-se o VCO tanque LC utilizando tecnologia TSMC 0.18 μ m, no qual alguns problemas em relação à ferramenta Cadence comprometeram a otimização do circuito. Por fim, a medida que cada bloco dos demais participantes do projeto do PLL ficaram prontos, foi realizada a simulação mista entre alguns blocos a nível de circuito e aqueles do modelo desenvolvido previamente em Verilog-AMS, validando-os a nível de sistema, segundo a metodologia de projeto *Top-Down*. Vale ressaltar que não foi possível realizar simulação mista com os blocos divisor de frequência e VCO, devido à limitação de memória das máquinas usadas. A partir das simulações mistas também foi possível fazer o levantamento do consumo de potência total do PLL, igual a 11.39 mW, valor razoável frente aos pesquisados para a mesma tecnologia e aplicação.

Portanto, pode-se dizer que os objetivos deste trabalho foram alcançados. Sua modelagem foi realizada com sucesso, o projeto do VCO gerou resultados razoáveis e foi possível validar, através de simulações mistas, a maioria dos blocos do PLL a nível de circuito, dando um bom ponto de partida à futura prototipação do sistema.

11.1 Trabalhos Futuros

Como trabalhos futuros, tem-se a otimização do VCO, a fim de reduzir o ruído de fase, além de projetar o estágio *buffer* de saída do mesmo; a simulação e otimização dos blocos a nível de circuito, através da validação de *corners* e Monte Carlo; a geração do *layout* e simulação pós-*layout* de todos os blocos; e, por fim, a prototipação do PLL para rodada de fabricação da tecnologia TSMC 0.18 μm .

Referências

- ANJOS, A. d. *Integração de blocos RF CMOS com indutores usando tecnologia Flip Chip*. Tese (Doutorado) — Universidade de São Paulo, 2012. Citado 2 vezes nas páginas 15 e 65.
- ARGÜELLO, A. M. G. Estudo e projeto de um sintetizador de frequência para rf em tecnologia cmos de $0.35\mu\text{m}$. 2004. Citado 4 vezes nas páginas 15, 52, 53 e 54.
- BANERJEE, D. *PLL performance, simulation and design*. [S.l.]: Dog Ear Publishing, 2006. Citado 4 vezes nas páginas 19, 55, 56 e 58.
- BARRETT, C. Fractional/integer-n pll basics. Citeseer, 1999. Citado 3 vezes nas páginas 49, 52 e 53.
- BATISTA, N.; MELÍCIO, R.; MENDES, V. Layered smart grid architecture approach and field tests by zigbee technology. *Energy Conversion and Management*, Elsevier, v. 88, p. 49–59, 2014. Citado 2 vezes nas páginas 15 e 35.
- BERNY, A. D. et al. *Analysis and design of wideband LC VCOs*. Tese (Doutorado) — University of California, Berkeley, 2006. Citado 2 vezes nas páginas 15 e 59.
- BILGIN, B. E.; GUNGOR, V. Performance evaluations of zigbee in different smart grid environments. *Computer Networks*, Elsevier, v. 56, n. 8, p. 2196–2205, 2012. Citado 2 vezes nas páginas 34 e 35.
- BISTUE, G.; QUEMADA, C.; ADIN, I. *Design methodology for RF CMOS phase locked loops*. [S.l.]: Artech House, 2009. Citado 14 vezes nas páginas 15, 16, 50, 54, 55, 56, 57, 62, 63, 64, 65, 66, 67 e 79.
- COELHO, C. *A tecnologia ZigBee*. 2013. Disponível em: <<http://www.taskblog.com.br/04/a-tecnologia-zigbee>>. Citado 2 vezes nas páginas 15 e 33.
- CORREA, F.; PAOLO, D. *Malha síncrona digital*. Tese (Doutorado) — Universidade de São Paulo, 2011. Citado na página 49.
- DABHI, R. A.; NAGPARA, B. H. 2ghz pll frequency synthesizer for zigbee applications. 2014. Citado 2 vezes nas páginas 15 e 50.
- DEVIDAS, A. R.; RAMESH, M. V. Wireless smart grid design for monitoring and optimizing electric transmission in india. In: IEEE. *Sensor Technologies and Applications (SENSORCOMM), 2010 Fourth International Conference on*. [S.l.], 2010. p. 637–640. Citado na página 35.
- ERGEN, S. C. Zigbee/ieee 802.15. 4 summary. *UC Berkeley, September*, v. 10, 2004. Citado 2 vezes nas páginas 19 e 32.
- FARFÁN, A. *Projeto e Implementação de um Oscilador Monolítico a 2, 4 GHz em Tecnologia CMOS $0.35\mu\text{m}$* . Tese (Doutorado) — Dissertação de Mestrado-Escola Politécnica, Universidade de São Paulo, São Paulo, 2003. Citado 6 vezes nas páginas 15, 16, 59, 61, 62 e 66.

- FERREIRA, F. J. A. *Projeto de um Misturador em Tecnologia CMOS de*. Tese (Doutorado) — Universidade Federal de Itajubá, 2006. Citado 2 vezes nas páginas 15 e 34.
- GOMES, W. de J. Projetos dos blocos charge pump, loop filter e pfd do phase locked loop de um transceptor zigbee. 2015. Citado 4 vezes nas páginas 91, 94, 96 e 119.
- HAMEL, J. S. Lc tank voltage controlled oscillator tutorial. *Waterloo, Ontario, Canada*, 2005. Citado na página 71.
- HENZLER, S. *High-Speed Digital CMOS Circuits : Phase Locked Loops (PLL)*. Tese (Doutorado) — Technische Universität München, 2011. Citado 3 vezes nas páginas 15, 51 e 60.
- INSTRUMENTS, T. Cc2420: 2.4 ghz ieee 802.15. 4/zigbee-ready rf transceiver. *Available at Available at <http://www.ti.com/lit/gpn/cc2420>*, 2006. Citado na página 85.
- ISMAIL, N. M. H.; OTHMAN, M. System analysis of 2.4 ghz ieee 802.15.4 compliant frequency synthesizer,. 2009. Citado na página 84.
- JOHANN, M. Estrutura de roteamento em circuitos vlsi. *Porto Alegre: CPGCC da UFRGS*, 1997. Citado 2 vezes nas páginas 15 e 43.
- KINGET, P. Integrated ghz voltage controlled oscillators. In: *Analog circuit design*. [S.l.]: Springer, 1999. p. 353–381. Citado na página 60.
- KUNDERT, K.; CHANG, H. Top-down design and verification of mixed-signal circuits. *www.designers-guide.com*, 2005. Citado 5 vezes nas páginas 41, 42, 45, 46 e 47.
- LEESON, D. B. A simple model of feedback oscillator noise spectrum. *Proceedings of the IEEE*, IEEE, v. 54, n. 2, p. 329–330, 1966. Citado na página 66.
- MADUREIRA, H. Projeto de oscilador controlado por tensão para transceptor rf 900mhz embarcado em soc. *Trabalho de Graduação do Departamento de Eng. Elétrica, Universidade de Brasília:[sn]*, 2008. Citado 3 vezes nas páginas 59, 61 e 76.
- MANTHENA, V. K. *Ultra Low Power CMOS Phase-Locked Loop Frequency Synthesizers*. Tese (Doutorado) — Nanyang Technological University, 2011. Citado 2 vezes nas páginas 15 e 57.
- MELNIK, D. *Verilog-AMS & Multi-Level Simulation – Aldec and Tanner EDA Bridge Digital and Analog Design Flows*. 2006. Disponível em: <<https://www.aldec.com/en/company/blog/50-verilog-ams-and-multi-level-simulation>>. Citado 3 vezes nas páginas 15, 45 e 46.
- MONSIGNORE, F. *Sensoriamento de ambiente utilizando o padrao ZigBee*. Tese (Doutorado) — Universidade de São Paulo, 2007. Citado na página 32.
- MOON, S. T. *Design of high performance frequency synthesizers in communication systems*. Tese (Doutorado) — Texas A&M University, 2005. Citado na página 86.
- NONAKA, I. Toward middle-up-down management: accelerating information creation. *Sloan management review*, v. 29, n. 3, p. 9–18, 1988. Citado na página 41.

- NORRIS, M. Single-chip zigbee for indoor mobile telemetry. IET, 2005. Citado na página 31.
- OGATA, K.; YANG, Y. Modern control engineering. Prentice-Hall Englewood Cliffs, 1970. Citado 2 vezes nas páginas 81 e 85.
- OH, N.-J.; LEE, S.-G. Building a 2.4-ghz radio transceiver using ieee 802.15. 4. *Circuits and Devices Magazine, IEEE*, IEEE, v. 21, n. 6, p. 43–51, 2006. Citado 2 vezes nas páginas 19 e 80.
- PINTO, J. A. de A. Projeto e modelagem de um divisor de frequências para utilização no pll de um transceptor zigbee. 2015. Citado 6 vezes nas páginas 102, 104, 119, 133, 135 e 137.
- RAZAVI, B.; BEHZAD, R. *RF microelectronics*. [S.l.]: Prentice Hall New Jersey, 1998. Citado 6 vezes nas páginas 15, 51, 53, 59, 60 e 81.
- SHU, K.; SÁNCHEZ-SINENCIO, E. *CMOS PLL Synthesizers: Analysis and Design: Analysis and Design*. [S.l.]: Springer, 2006. Citado na página 84.
- SILVA, I. M. D. da. *Análise de desempenho de sistemas de comunicação sem-fio para monitoramento de unidade de produção de poços petrolíferos terrestres*. Tese (Doutorado) — Universidade Federal do Rio Grande do Norte, 2008. Citado 2 vezes nas páginas 15 e 32.
- SRINIVASAN, R. *Design and implementation of a frequency synthesizer for an IEEE 802.15. 4/Zigbee transceiver*. Tese (Doutorado) — Texas A&M University, 2006. Citado 5 vezes nas páginas 16, 81, 82, 83 e 84.
- VALERO-LOPEZ, A. Y. *Design of Frequency Synthesizers for Short-Range Wireless Systems*. Tese (Doutorado) — Texas A&M University, 2004. Citado na página 86.
- ZURITA, M. *Metodologia e Fluxo de Projeto de Sistemas VLSI digitais*. Tese (Doutorado) — Universidade Federal do Piauí, 2013. Citado 3 vezes nas páginas 15, 41 e 42.

Apêndices

APÊNDICE A – Simulações Adicionais

A.1 Divisor de Frequências

Main Counter

O *Main Counter* é um contador assíncrono utilizado como divisor de frequências por 32, composto por 5 *flip-flops* D, na configuração de divisor por 2, em cascata. Na sequência, encontram-se o esquemático, o *testbench*, a pinagem e a simulação. Na simulação é possível ver a saída de cada um dos 5 *flip-flops* D configurados como divisor por 2, de forma que a cada estágio a divisão segue o padrão 2^n , onde n é o estágio começando de 0 até quantidade de estágios menos 1. Para mais informações sobre o bloco, [Pinto \(2015\)](#).

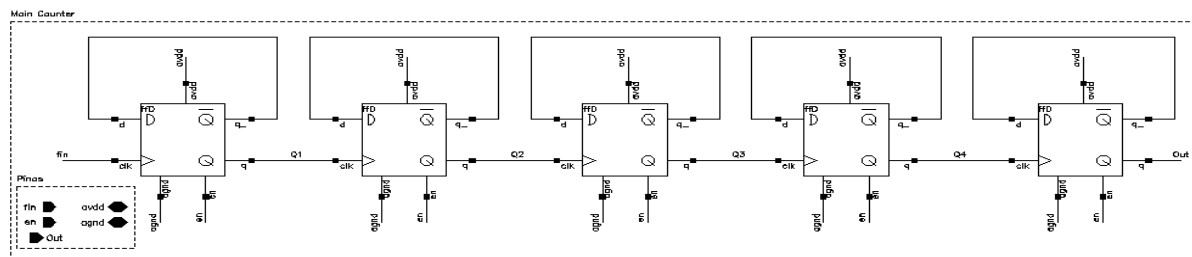


Figura 76 – Esquemático do *Main Counter*.

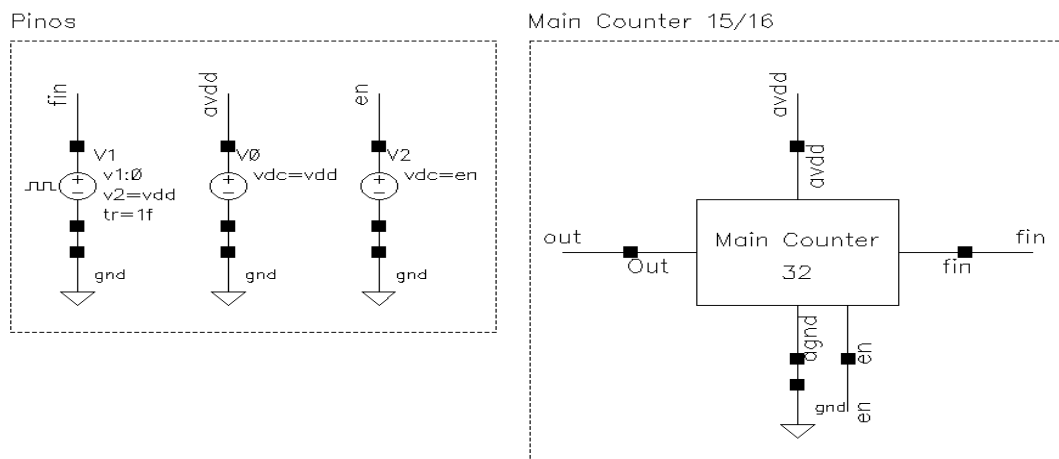
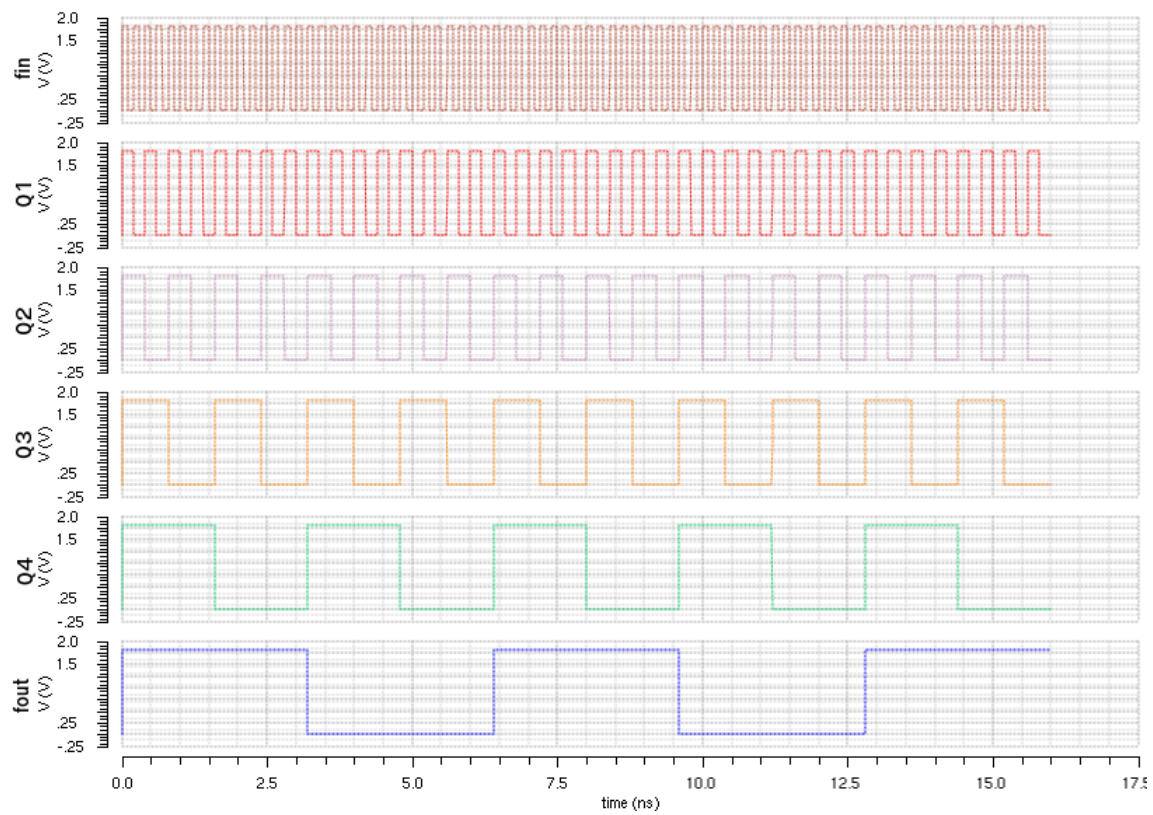


Figura 77 – *Testbench* do *Main Counter*.

Tabela 24 – Descrição dos pinos do *Main Counter*.

Pino	Descrição	Tipo
F_{In}	Entrada	<i>input</i>
F_{Out}	Saída	<i>output</i>
avdd	Tensão de alimentação	<i>inout</i>
agnd	Tensão de referência (<i>ground</i>)	<i>inout</i>
en	Habilitar (<i>enable</i>)	<i>input</i>

Figura 78 – Simulação do *Main Counter*.

Prescaler

O bloco *Prescaler*, assim como o *Main Counter*, é um divisor de frequências, no entanto, este bloco possui uma chave binária (MC) usada para seleção do fator de divisão. Nesta aplicação, a topologia escolhida possui os fatores de divisão 15 e 16, onde caso MC esteja em nível baixo, a saída consiste na onda de entrada do bloco com frequência dividida por 15, e para MC em nível alto, a frequência é dividida por 16. Na sequência, encontram-se o esquemático, o *testbench*, a pinagem e as simulações que demonstram o comportamento explicitado acima. Para mais informações sobre o bloco, [Pinto \(2015\)](#).

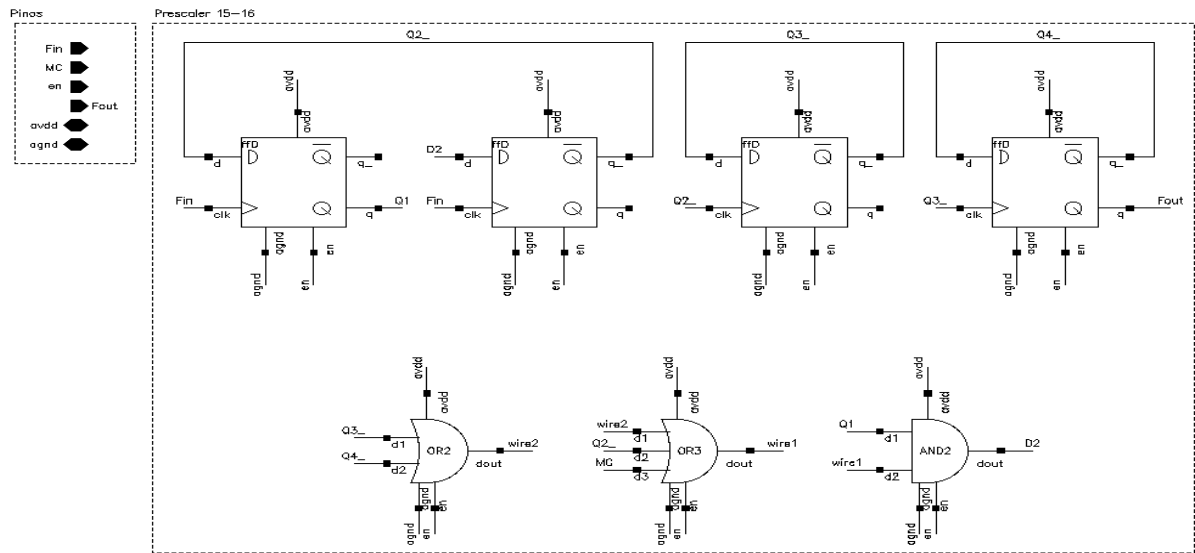


Figura 79 – Esquemático do *Prescaler*.

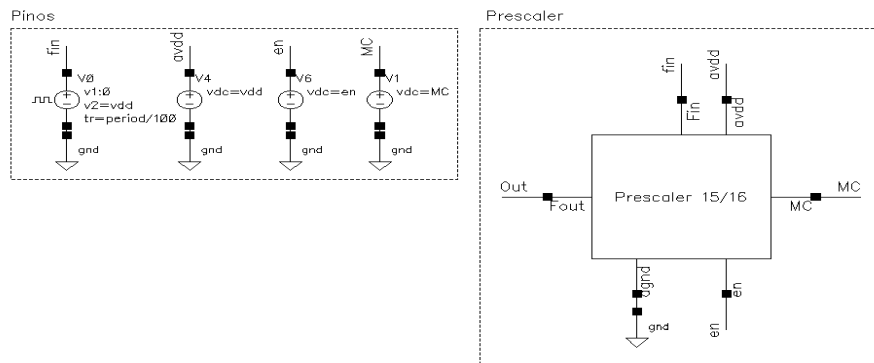
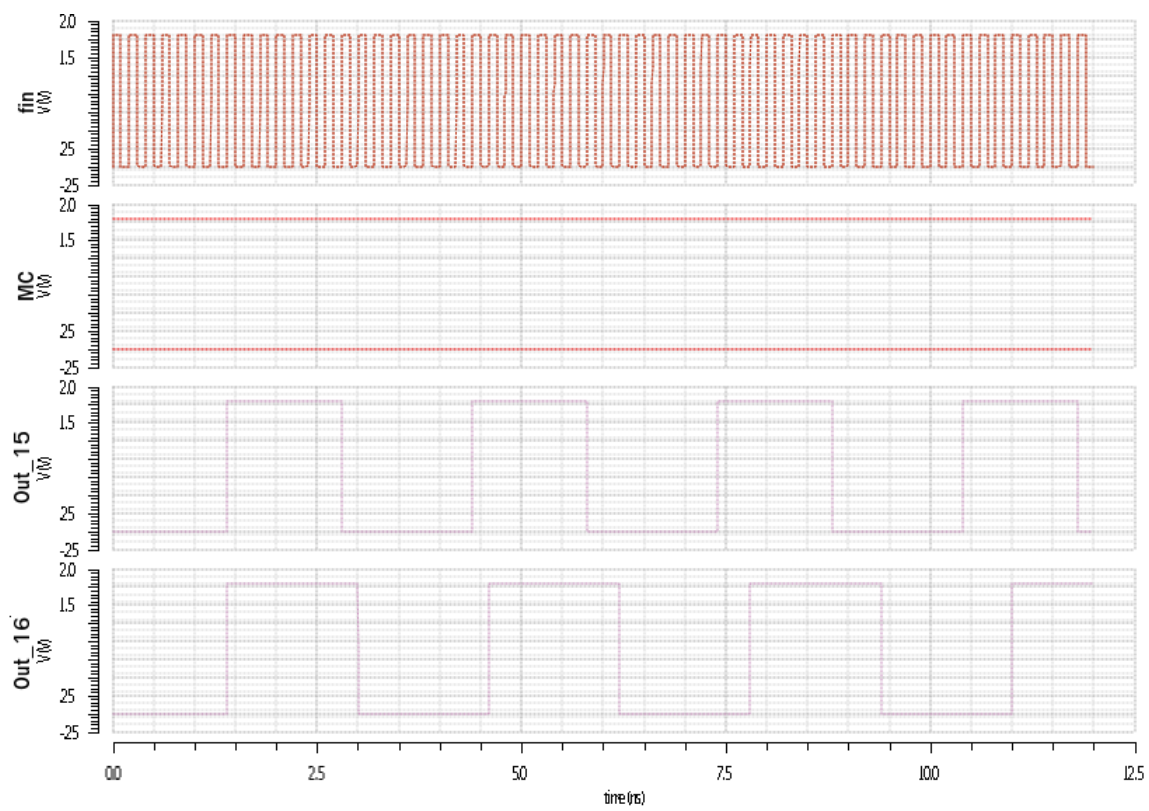
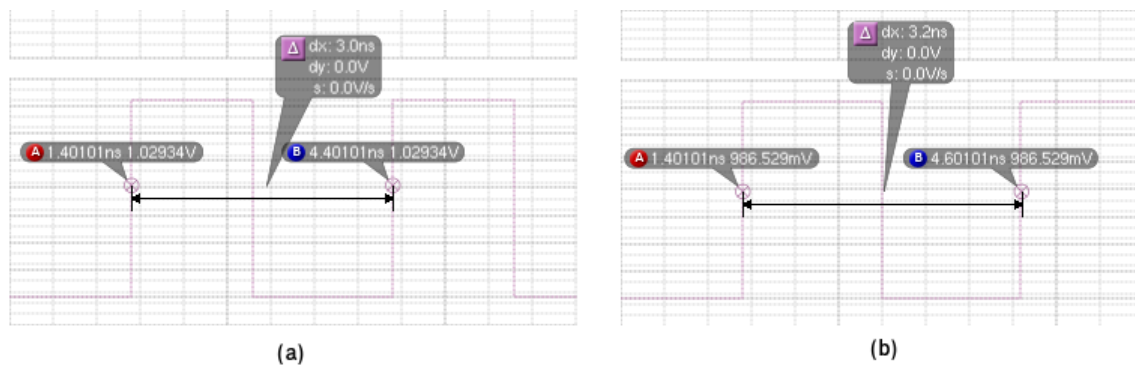


Figura 80 – *Testbench* do *Prescaler*.

Tabela 25 – Descrição dos pinos do *Prescaler*.

Pino	Descrição	Tipo
F_{In}	Entrada	<i>input</i>
MC	Seleção do fator de divisão	<i>input</i>
F_{Out}	Saída	<i>output</i>
avdd	Tensão de alimentação	<i>inout</i>
agnd	Tensão de referência (<i>ground</i>)	<i>inout</i>
en	Habilitar (<i>enable</i>)	<i>input</i>

Figura 81 – Simulação do *Prescaler*.Figura 82 – Fatores de divisão do *Prescaler*: (a) 15; (b) 16.

Scounter

Scounter ou *Swallow Counter* é basicamente um contador cíclico decrescente. No entanto, este possui uma configuração para setar o valor inicial da contagem, configuração esta ditada pela entrada *_lo*. Nesta aplicação, o valor inicial é composto por uma palavra de 4 *bits*, onde a partir do momento que *_lo* encontra-se em nível baixo, a saída estabiliza no valor da palavra S, por outro lado, no momento em que *_lo* está alto, a saída do *Scounter* começa a contagem decrescente a partir do valor de S. Caso a contagem chegue no valor 0000, o próximos valores são 1111, 1110, etc. Na sequência, encontram-se o esquemático, o *testbench*, a pinagem e a simulação. Para mais informações, [Pinto \(2015\)](#).

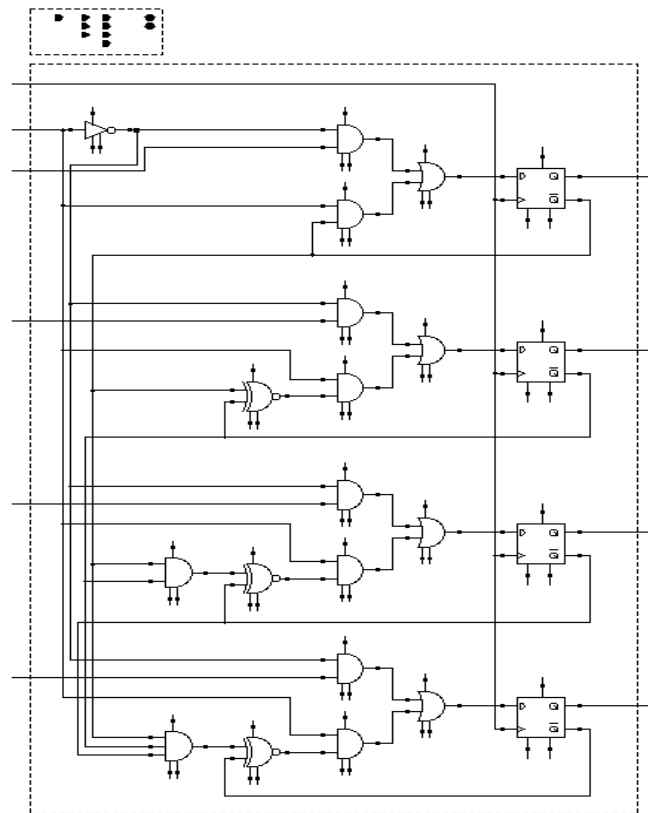


Figura 83 – Esquemático do *Scounter*.

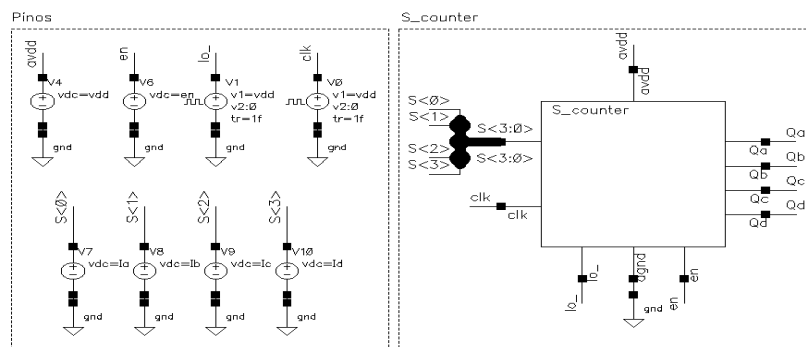
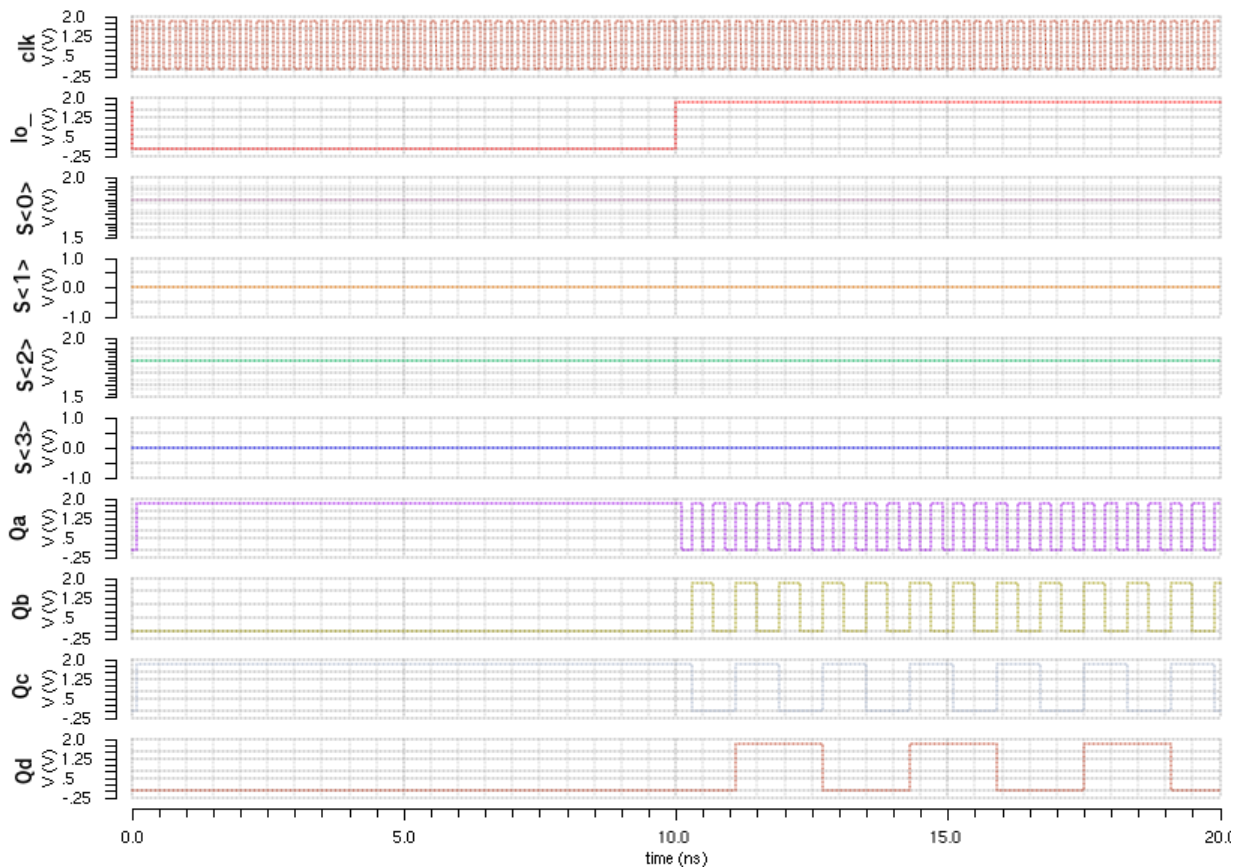


Figura 84 – *Testbench* do *Scounter*.

Tabela 26 – Descrição dos pinos do *Scounter*.

Pino	Descrição	Tipo
F_{In}	Entrada	<i>input</i>
$S < 4 >$	Vetor de entrada digital para setar estado inicial do contador	<i>input</i>
lo_	Entrada para habilitar estado inicial do contador	<i>input</i>
Q_a	Saída Q_a	<i>output</i>
Q_b	Saída Q_b	<i>output</i>
Q_c	Saída Q_c	<i>output</i>
Q_d	Saída Q_d	<i>output</i>
avdd	Tensão de alimentação	<i>inout</i>
agnd	Tensão de referência (<i>ground</i>)	<i>inout</i>
en	Habilitar (<i>enable</i>)	<i>input</i>

Figura 85 – Simulação do *Scounter*.

APÊNDICE B – Códigos da Modelagem em Verilog-AMS

B.1 Detector de Fase e Frequência (PFD)

```

/*
PFD - Phase/Frequency Detector
Detector de Fase e Frequencia

Autor: Thiago Almeida Nunes Guimaraes
Matricula: 09/0133641
*/

//Bibliotecas
`include "constants.vams"
`include "disciplines.vams"

//Modulo PFD_v2
module PFD_v2(UP, DOWN, fref, fdiv, avdd, agnd, en);

//Pinos
output UP, DOWN;
inout avdd, agnd;
input fref, fdiv, en;

//Tipos de sinais
electrical UP, DOWN, fref, fdiv, avdd, agnd, en;

//Parametros
parameter real vth = 0.9,
              td = 1f from (0:inf),
              tt = 1f from [0:inf);

//Variaveis internas
real enable;
integer dig1, dig2, reset;

//Processo analogico
analog begin: main

//Configurando enable
enable = (V(en) > vth) ? 0 : 1;

@(initial_step) begin
    dig1 = 0;
    dig2 = 0;
end

//Detectando threshold
@(cross(V(fref) - vth, 1) or posedge(reset));
    reset = dig1 && dig2;
    dig1 = (V(fref) > vth);

```

```

    @(cross(V(fdiv) - vth, 1) or posedge(reset));
        reset = dig1 && dig2;
        dig2 = (V(fdiv) > vth) && (dig1 > vth);

    //Gerando sinal de saída
    V(UP) <+ transition(V(avdd)*dig1*!reset*(enable), td, tt);
    V(DOWN) <+ transition(V(avdd)*dig2*!reset*(enable), td, tt);

end
endmodule

```

B.2 Charge Pump (CP)

```

/*
    CP - Charge Pump

    Autor: Thiago Almeida Nunes Guimaraes
    Matricula: 09/0133641
*/

//Bibliotecas
`include "constants.vams"
`include "disciplines.vams"

//Modulo CP
module CP (aout, up, down, avdd, agnd, en);

    //Pinos
    output aout;
    inout avdd, agnd;
    input up, down, en;

    //Tipos de sinais
    electrical aout, up, down, avdd, agnd, en;

    //Parametros
    parameter real vth = 0.9,
                cur = 20u,
                td = 1f from (0:inf),
                tt = 1f from [0:inf);

    //Variaveis internas
    real aux, enable;

    //Processo analogico
    analog begin: main

        //Valores iniciais
        @(initial_step) aux = 0.0;

        //Configurando enable
        enable = ((V(en) > vth) ? 0 : 1);
    end
endmodule

```

```

//CP
if(V(down)>vth && V(up)<vth)    aux = -cur;
else if(V(down)<vth && V(up)>vth) aux = cur;
else                            aux = 0;

//Gerando sinal de saida
I(aout) <+ transition(aux*enable, td, tt);

end
endmodule

```

B.3 Filtro de Malha

```

/*
  Filtro de Malha (Loop Filter)

  Autor: Thiago Almeida Nunes Guimaraes
  Matricula: 09/0133641
*/

//Bibliotecas
#include "constants.vams"
#include "disciplines.vams"

//Modulo LF
module LF_v1(p, n, en);

  //Pinos
  inout p, n;
  input en;

  //Tipo de sinais
  electrical p, n, i, en;

  //Parametros
  parameter real vth = 0.9,
               r1 = 250K,
               c1 = 16p,
               c2 = 1p;

  //Branches
  branch (p, i) res1;
  branch (i, n) cap1;
  branch (p, n) cap2;

  //Variaveis internas
  real enable;

  //Processo analogico
  analog begin: main

    //Configurando enable
    enable = ((V(en) > vth) ? 0 : 1);

```

```

        //Gerando sinais de saída
        V(res1) <+ r1*I(res1);
        I(cap1) <+ c1*ddt(V(cap1));
        I(cap2) <+ c2*ddt(V(cap2));

    end
endmodule

```

B.4 Oscilador Controlado por Tensão (VCO)

```

/*
    VCO - Voltage Controlled Oscillator
    Oscilador Controlado por Tensao

    Autor: Thiago Almeida Nunes Guimaraes
    Matricula: 09/0133641
*/

//Bibliotecas
`include "constants.vams"
`include "disciplines.vams"

//Modulo VCO
module VCO(aout_n, aout_p, ain, avdd, agnd, en);

    //Pinos
    output aout_n, aout_p;
    inout avdd, agnd;
    input ain, en;

    //Tipos de sinais
    electrical aout_n, aout_p, ain, avdd, agnd, en;

    //Parametros
    parameter real vth = 0.9,
                  Vmin = 0,
                  Vmax = Vmin + 1 from (Vmin:inf),
                  Fmin = 4.81G from (0:inf),
                  Fmax = Fmin + 2*(15*5M) from (Fmin:inf),
                  ampl = 0.45;

    //Variaveis internas
    real freq, phase, enable;

    //Processo analogico
    analog begin: main

        //Configurando enable
        enable = (V(en) > vth) ? 0 : 1;

        //Calculo da frequencia a partir de uma tensao de entrada
        freq = (V(ain) - Vmin) * (Fmax - Fmin) / (Vmax - Vmin) + Fmin;

        //Limitando a frequencia
        if (freq > Fmax) freq = Fmax;
    end
endmodule

```



```

    if(freq < Fmin)    freq = Fmin;

    //Calculo da fase, a fase e a integral de freq modulo 2pi
    phase = 2*M_PI*idtmod(freq, 0.0, 1.0, -0.5);

    //Gerando os sinais diferenciais de saida
    V(aout_p) <+ (((ampl*sin(phase))/2) + (V(avdd)/2)) * (enable);
    V(aout_n) <+ ((-(ampl*sin(phase))/2) + (V(avdd)/2)) * (enable);

    //Ajustando o passo de tempo
    $bound_step(0.1/freq);

end
endmodule

```

B.5 Conversor de Saída Diferencial para *Single Ended*

```

/*
    Conversor de saida diferencial para single ended, saturando a saida

    Autor: Thiago Almeida Nunes Guimaraes
    Matricula: 09/0133641
*/

//Bibliotecas
`include "constants.vams"
`include "disciplines.vams"

//Modulo diff2single
module diff2single (dout, ain_n, ain_p, avdd, agnd, en);

    //Pinos
    output dout;
    inout avdd, agnd;
    input ain_n, ain_p, en;

    //Tipos de sinais
    electrical dout, ain_n, ain_p, avdd, agnd, en;

    //Parametros
    parameter real vth = 0.9,
                td = 1f from (0:inf),
                tt = 1f from [0:inf);

    //Variaveis internas
    real enable, vout, voutd;

    //Processo analogico
    analog begin: main

        //Configurando enable
        enable = ((V(en) > vth) ? 0 : 1);

        //Convertendo de diferencial para single ended
        vout = ((V(ain_p) - V(ain_n)) + V(avdd)/2);

```

```
//Detectando threshold
@(cross(vout - vth));

//Convertendo sinal analogico para digital
voutd = ((vout > vth) ? V(avdd) : V(agnd));

//Gerando o sinal de saida
V(dout) <+ transition(voutd*(enable), td, tt);

end
endmodule
```

B.6 Portas Lógicas

Inversora

```
/*
    Porta NOT - inversora

    Autor: Thiago Almeida Nunes Guimaraes
    Matricula: 09/0133641
*/

//Bibliotecas
#include "constants.vams"
#include "disciplines.vams"

//Modulo NOT
module NOT(dout, d1, avdd, agnd, en);

    //Pinos
    output dout;
    inout avdd, agnd;
    input d1, en;

    //Tipos de sinais
    electrical dout, d1, avdd, agnd, en;

    //Parametros
    parameter real vth = 0.9,
                td = 1f from (0:inf),
                tt = 1f from [0:inf);

    //Variaveis internas
    real vout, enable;
    integer dig1;

    //Processo analogico
    analog begin: main

        //Configurando enable
        enable = (V(en) > vth) ? 0 : 1;

        //Detectando threshold
        @(cross(V(d1) - vth));
        dig1 = (V(d1) > vth);

        //Gerando sinal de saida
        vout = (!dig1) ? V(avdd) : V(agnd);
        V(dout) <+ transition(vout*enable, td, tt);

    end
endmodule
```

Buffer

```
/*
    Buffer

    Autor: Thiago Almeida Nunes Guimaraes
    Matricula: 09/0133641
*/

//Bibliotecas
`include "constants.vams"
`include "disciplines.vams"

//Modulo PFD_buffer
module PFD_buffer (Out, In);

    //Pinos
    output Out;
    input In;

    //Tipos de sinais
    electrical Out, In;

    //Parametros
    parameter real delay = 10p;

    //Processo analogico
    analog begin: main

        //Gerando sinais de saida
        V(Out) <+ absdelay(V(In), delay);

    end
endmodule
```

AND2

```
/*
  Porta AND de duas entradas

  Autor: Thiago Almeida Nunes Guimaraes
  Matricula: 09/0133641
*/

//Bibliotecas
`include "constants.vams"
`include "disciplines.vams"

//Modulo AND2
module AND2(dout, d1, d2, avdd, agnd, en);

  //Pinos
  output dout;
  inout avdd, agnd;
  input d1, d2, en;

  //Tipos de sinais
  electrical dout, d1, d2, avdd, agnd, en;

  //Parametros
  parameter real vth = 0.9,
               td = 1f from (0:inf),
               tt = 1f from [0:inf);

  //Variaveis internas
  real vout, enable;
  integer dig1, dig2;

  //Processo analogico
  analog begin: main

    //Configurando enable
    enable = (V(en) > vth) ? 0 : 1;

    //Detectando threshold
    @(cross(V(d1) - vth) or cross(V(d2) - vth));
    dig1 = (V(d1) > vth);
    dig2 = (V(d2) > vth);

    //Gerando sinal de saida
    vout = (dig1 && dig2) ? V(avdd) : V(agnd);
    V(dout) <+ transition(vout*enable, td, tt);

  end
endmodule
```

AND3

```

/*
    Porta AND de tres entradas

    Autor: Thiago Almeida Nunes Guimaraes
    Matricula: 09/0133641
*/

//Bibliotecas
`include "constants.vams"
`include "disciplines.vams"

//Modulo AND3
module AND3(dout, d1, d2, d3, avdd, agnd, en);

    //Pinos
    output dout;
    inout avdd, agnd;
    input d1, d2, d3, en;

    //Tipos de sinais
    electrical dout, d1, d2, d3, avdd, agnd, en;

    //Parametros
    parameter real vth = 0.9,
                  td = 1f from (0:inf),
                  tt = 1f from [0:inf);

    //Variaveis internas
    real vout, enable;
    integer dig1, dig2, dig3;

    //Processo analogico
    analog begin: main

        //Configurando enable
        enable = (V(en) > vth) ? 0 : 1;

        //Detectando threshold
        @(cross(V(d1) - vth) or cross(V(d2) - vth) or cross(V(d3) - vth));
        dig1 = (V(d1) > vth);
        dig2 = (V(d2) > vth);
        dig3 = (V(d3) > vth);

        //Gerando sinal de saida
        vout = ((dig1 && dig2 && dig3) ? V(avdd) : V(agnd));
        V(dout) <+ transition(vout*enable, td, tt);

    end
endmodule

```

OR2

```
/*
  Porta OR de duas entradas

  Autor: Thiago Almeida Nunes Guimaraes
  Matricula: 09/0133641
*/

//Bibliotecas
#include "constants.vams"
#include "disciplines.vams"

//Modulo OR2
module OR2(dout, d1, d2, avdd, agnd, en);

  //Pinos
  output dout;
  inout avdd, agnd;
  input d1, d2, en;

  //Tipos de sinais
  electrical dout, d1, d2, avdd, agnd, en;

  //Parametros
  parameter real vth = 0.9,
               td = 1f from (0:inf),
               tt = 1f from [0:inf);

  //Variaveis internas
  real vout, enable;
  integer dig1, dig2;

  //Processo analogico
  analog begin: main

    //Configurando enable
    enable = (V(en) > vth) ? 0 : 1;

    //Detectando threshold
    @(cross(V(d1) - vth) or cross(V(d2) - vth));
    dig1 = (V(d1) > vth);
    dig2 = (V(d2) > vth);

    //Gerando sinal de saida
    vout = (dig1 || dig2) ? V(avdd) : V(agnd);
    V(dout) <+ transition(vout*enable, td, tt);

  end
endmodule
```

OR3

```
/*
    Porta OR de tres entradas

    Autor: Thiago Almeida Nunes Guimaraes
    Matricula: 09/0133641
*/

//Bibliotecas
`include "constants.vams"
`include "disciplines.vams"

//Modulo OR3
module OR3(dout, d1, d2, d3, avdd, agnd, en);

    //Pinos
    output dout;
    inout avdd, agnd;
    input d1, d2, d3, en;

    //Tipos de sinais
    electrical dout, d1, d2, d3, avdd, agnd, en;

    //Parametros
    parameter real vth = 0.9,
                td = 1f from (0:inf),
                tt = 1f from [0:inf);

    //Variaveis internas
    real vout, enable;
    integer dig1, dig2, dig3;

    //Processo analogico
    analog begin: main

        //Configurando enable
        enable = (V(en) > vth) ? 0 : 1;

        //Detectando threshold
        @(cross(V(d1) - vth) or cross(V(d2) - vth));
        dig1 = (V(d1) > vth);
        dig2 = (V(d2) > vth);
        dig3 = (V(d3) > vth);

        //Gerando sinal de saida
        vout = (dig1 || dig2 || dig3) ? V(avdd) : V(agnd);
        V(dout) <+ transition(vout*enable, td, tt);

    end
endmodule
```


OR4

```
/*
  Porta OR de quatro entradas

  Autor: Thiago Almeida Nunes Guimaraes
  Matricula: 09/0133641
*/

//Bibliotecas
#include "constants.vams"
#include "disciplines.vams"

//Modulo OR4
module OR4(dout, d1, d2, d3, d4, avdd, agnd, en);

  //Pinos
  output dout;
  inout avdd, agnd;
  input d1, d2, d3, d4, en;

  //Tipos de sinais
  electrical dout, d1, d2, d3, d4, avdd, agnd, en;

  //Parametros
  parameter real vth = 0.9,
               td = 1f from (0:inf),
               tt = 1f from [0:inf);

  //Variaveis internas
  real vout, enable;
  integer dig1, dig2, dig3, dig4;

  //Processo analogico
  analog begin: main

    //Configurando enable
    enable = (V(en) > vth) ? 0 : 1;

    //Detectando threshold
    @(cross(V(d1) - vth) or cross(V(d2) - vth));
    dig1 = (V(d1) > vth);
    dig2 = (V(d2) > vth);
    dig3 = (V(d3) > vth);
    dig4 = (V(d4) > vth);

    //Gerando sinal de saida
    vout = (dig1 || dig2 || dig3 || dig4) ? V(avdd) : V(agnd);
    V(dout) <+ transition(vout*enable, td, tt);

  end
endmodule
```

XNOR2

```

/*
    Porta XNOR de duas entradas

    Autor: Thiago Almeida Nunes Guimaraes
    Matricula: 09/0133641
*/

//Bibliotecas
`include "constants.vams"
`include "disciplines.vams"

//Modulo XNOR2
module XNOR2(dout, d1, d2, avdd, agnd, en);

    //Pinos
    output dout;
    inout avdd, agnd;
    input d1, d2, en;

    //Tipos de sinais
    electrical dout, d1, d2, avdd, agnd, en;

    //Parametros
    parameter real vth = 0.9,
                  td = 1f from (0:inf),
                  tt = 1f from [0:inf);

    //Variaveis internas
    real vout, enable;
    integer dig1, dig2;

    //Processo analogico
    analog begin: main

        //Configurando enable
        enable = (V(en) > vth) ? 0 : 1;

        //Detectando threshold
        @(cross(V(d1) - vth) or cross(V(d2) - vth));
        dig1 = (V(d1) > vth);
        dig2 = (V(d2) > vth);

        //Gerando sinal de saida
        vout = (dig1 == dig2) ? V(avdd) : V(agnd);
        V(dout) <+ transition(vout*enable, td, tt);

    end
endmodule

```

B.7 Multiplexador

```
/*  
    Mux 2x1  
  
    Autor: Thiago Almeida Nunes Guimaraes  
    Matricula: 09/0133641  
*/  
  
//Bibliotecas  
`include "constants.vams"  
`include "disciplines.vams"  
  
//Modulo MUX2x1  
module MUX2x1(out, S, in1, in2);  
  
    //Pinos  
    output out;  
    input S, in1, in2;  
  
    //Tipos de sinais  
    electrical out, S, in1, in2;  
  
    //Parametros  
    parameter real vth = 0.9,  
                td = 1f from (0:inf),  
                tt = 1f from [0:inf);  
  
    //Variaveis internas  
    real vout;  
  
    //Processo analogico  
    analog begin: main  
  
        //Selecao  
        vout = (V(S) < vth) ? V(in1) : V(in2);  
  
        //Gerando sinais de saida  
        V(out) <+ transition(vout, td, tt);  
  
    end  
endmodule
```

B.8 *Flip-Flops*

Flip-Flip D

```

/*
    Flip - flop D

    Autor: Thiago Almeida Nunes Guimaraes
    Matricula: 09/0133641
*/

//Bibliotecas
`include "constants.vams"
`include "disciplines.vams"

//Modulo ffD
module ffD(q, q_, clk, d, avdd, agnd, en);

    //Pinos
    output q, q_;
    inout avdd, agnd;
    input d, clk, en;

    //Tipos de sinais
    electrical q, q_, clk, d, avdd, agnd, en;

    //Parametros
    parameter real vth = 0.9,
                td = 1f from (0:inf),
                tt = 1f from [0:inf);
    parameter integer dir = 1 from [-1:1] exclude 0;

    //Variaveis internas
    real enable;
    integer dig;

    //Processo analogico
    analog begin: main

        //Configurando enable
        enable = ((V(en) > vth) ? 0 : 1);

        //Setando configuracoes iniciais
        @(initial_step) dig = 0;

        //Detectando threshold
        @(cross(V(clk) - vth, dir))
            dig = (V(d) > vth);

        //Gerando sinais de saida
        V(q) <+ transition(V(avdd)*dig*(enable), td, tt);
        V(q_) <+ transition(V(avdd)*(!dig)*(enable), td, tt);

    end
endmodule

```

Flip-Flop D com reset ativo baixo

```
/*
    Flip-flop D com reset ativo baixo

    Autor: Thiago Almeida Nunes Guimaraes
    Matricula: 09/0133641
*/

//Bibliotecas
#include "constants.vams"
#include "disciplines.vams"

//Modulo fFD_w_R
module fFD_w_R(q, clk, d, avdd, agnd, reset);

    //Pinos
    output q;
    inout avdd, agnd;
    input d, clk, reset;

    //Tipos de sinais
    electrical q, clk, d, avdd, agnd, reset;

    //Parametros
    parameter real vth = 0.9,
                td = 1f from (0:inf),
                tt = 1f from [0:inf);
    parameter integer dir = 1 from [-1:1] exclude 0;

    //Variaveis internas
    integer dig, digaux, vreset;

    //Processo analogico
    analog begin: main

        //Setando configuracoes iniciais
        @(initial_step) begin
            dig = 0;
            vreset = 0;
            digaux = 0;
        end

        //Detectando threshold
        @(cross(V(reset) - vth))
            vreset = (V(reset) > vth);

        @(cross(V(clk) - vth, dir))
            dig = (V(d) > vth);

        digaux = (!vreset && dig);

        //Gerando sinais de saida
        V(q) <+ transition(V(avdd)*digaux, td, tt);

    end
endmodule
```


APÊNDICE C – Exemplo de Modelagem em Verilog-AMS utilizando ferramentas Cadence

A modelagem de um sistema em alto nível parte da abstração do bloco para a descrição do mesmo na linguagem HDL escolhida. No caso, será desenvolvido como exemplo, um conversor AD de 16 *bits* em Verilog-AMS utilizando ferramentas Cadence. A fim de objetividade, a abstração do funcionamento detalhado do sistema será omitido, pois o objetivo é mostrar os passos necessários para simulações em plataforma Cadence usando a linguagem Verilog-AMS.

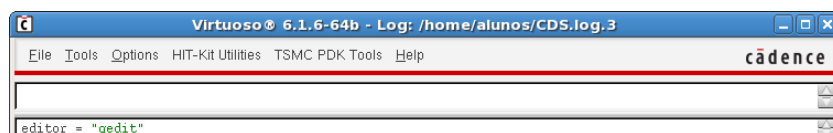


Figura 86 – Configuração do editor de texto usado para desenvolver código em Verilog-AMS.

Primeiramente, cria-se uma *library* para desenvolver o projeto em questão. A seguir, deve-se criar a *cellview*, Fig.(88-a), dedicada ao desenvolvimento do código em Verilog-AMS. Entretanto, um passo importante é a escolha do editor de textos a ser usado (Fig. 86). Uma vez o editor de texto escolhido, pode-se desenvolver o código.



Figura 87 – Criação da *library* para desenvolvimento do projeto.

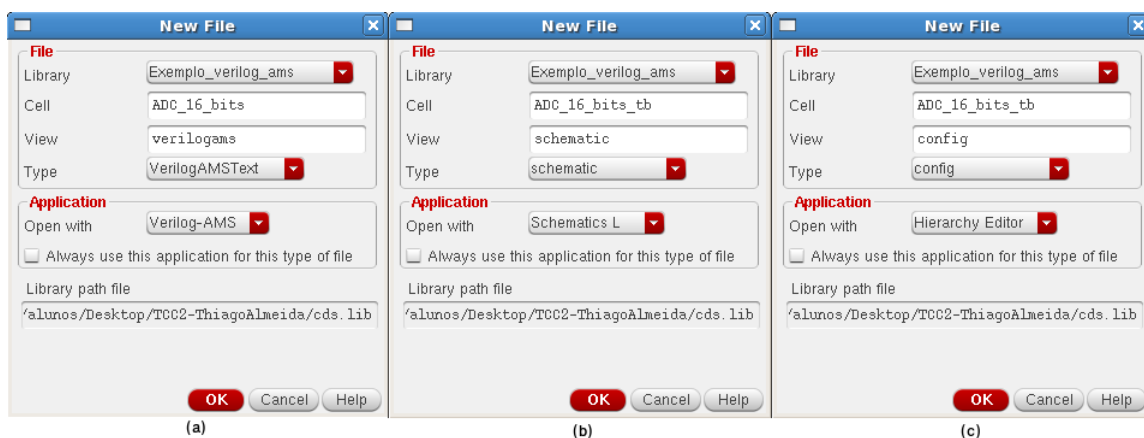


Figura 88 – Criação das *cellviews* necessárias para desenvolvimento do projeto: (a) Vista *verilogams*; (b) Vista *schematic*; (c) Vista *config*.

O código do conversor AD, encontra-se a seguir:

```

1  `include "constants.vams"
2  `include "disciplines.vams"
3
4  module ADCnbits(dout, ain, clk, avdd, agnd, en);
5      parameter real    vth = 0.9, vhigh = 1.8, vlow = 0.0, td = 1f, tt = 1f;
6      parameter integer bits = 16, dir = 1 from [-1:1] exclude 0;
7
8      output [bits-1:0] dout;
9      inout avdd, agnd;
10     input ain, clk, en;
11     real result [bits-1:0];
12     real sample, midpoint, enable;
13     genvar i;
14
15     electrical [bits-1:0] dout;
16     electrical ain, clk, en, avdd, agnd;
17
18     analog begin: main
19         enable = (V(en) > vth) ? 0 : 1;
20
21         @(cross(V(clk)-vth, dir) or initial_step) begin
22             sample = V(ain); midpoint = vhigh/2.0;
23             for(i=bits-1; i>=0; i=i-1) begin
24                 if(sample > midpoint) begin
25                     result[i] = vhigh; sample = sample - midpoint;
26                 end
27                 else
28                     result[i] = vlow; sample = 2.0 * sample;
29             end
30         end
31
32         for(i=bits-1; i>=0; i=i-1)
33             V(dout[i]) <+ transition(result[i]*(enable), td, tt);
34     end
35 endmodule

```

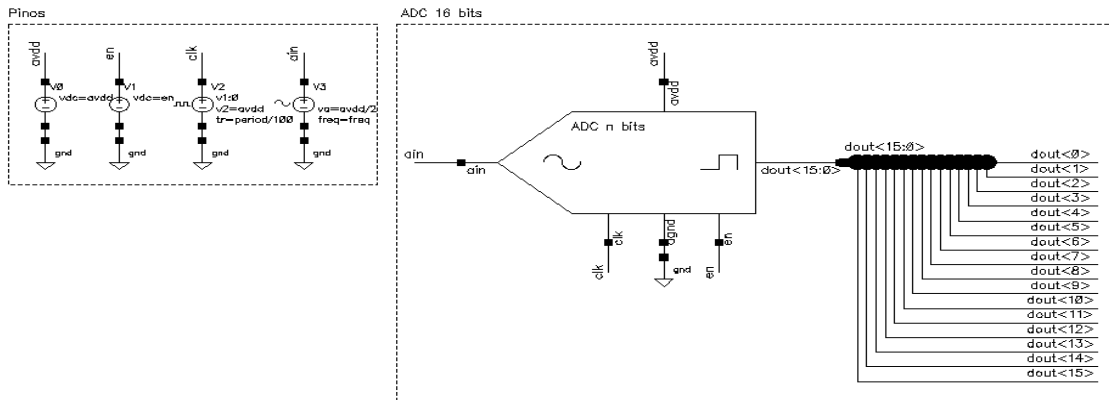


Figura 89 – Testbench do conversor AD de 16 bits modelado em Verilog-AMS, vista schematic.

Após escrito o código, para compilá-lo, basta fechar o editor de texto. Caso esteja tudo correto, duas mensagens aparecerão (Fig. 90): um de *warnings/errors* e uma referente a criação de um novo símbolo para a vista *verilogams* criada.

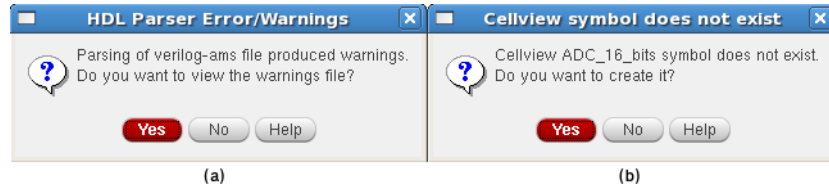


Figura 90 – Mensagens após compilação do código: (a) *Errors/Warnings*; (b) Criação do símbolo.

Posteriormente a criação do código e do símbolo referente ao mesmo, cria-se a vista *schematic* (Fig.88-b), esta por finalidade de testar o modelo descrito, conforme (Fig.89). Nota-se o uso de vetor no esquemático, *bus wire*.

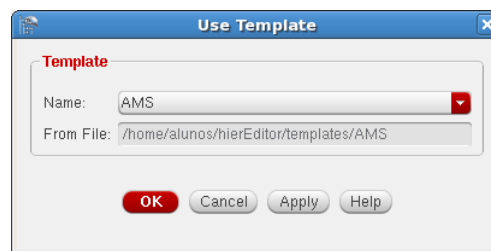


Figura 91 – *Template* usado na vista *config* do conversor AD.

O passo a seguir é a criação da vista *config*, Fig.(88-c), necessária para a simulação da vista *schematic* desenvolvida para teste. Ao criá-la, deve-se escolher o *template* AMS, confirmar, salvar e fechar a vista.

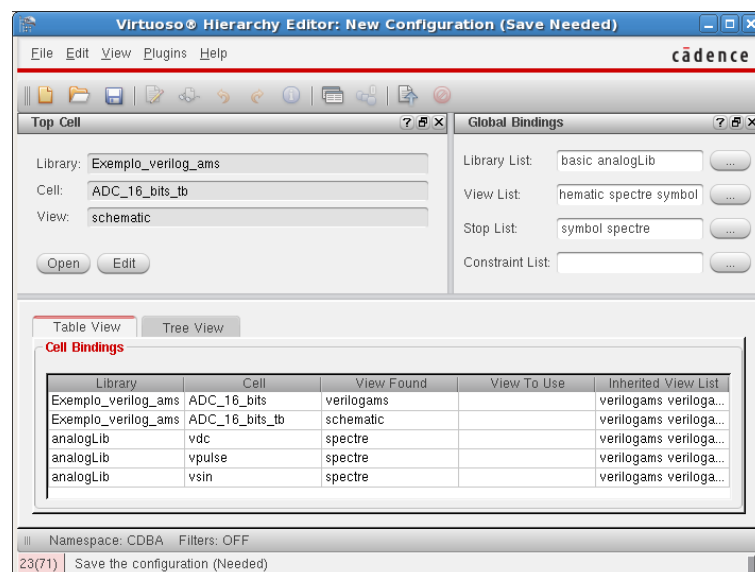


Figura 92 – Vista *config* do conversor AD.

Após fechar a vista *config*, deve-se abri-la novamente, o esquemático do *testbench* criado anteriormente será aberto junto com a mesma. A partir deste ponto, é possível criar simulações normalmente utilizando o ADE L e/ou configurar simulações para validação de *corners* no ADE XL. A única configuração a ser feita antes da simulação é a mudança do simulador utilizado, escolhe-se o AMS.

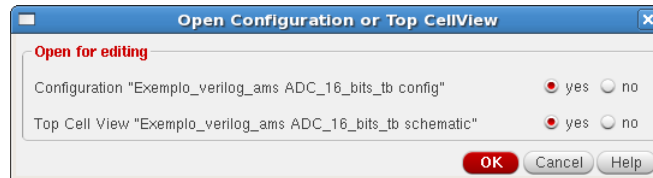


Figura 93 – Abertura da vista *config*.

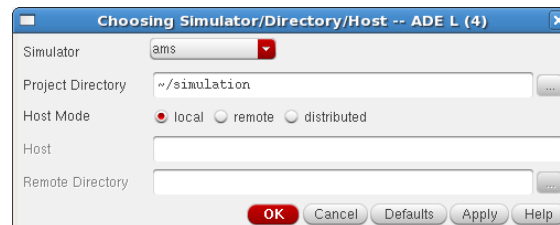


Figura 94 – Escolha do simulador AMS para simulações no ADE L.

Um aspecto interessante da linguagem Verilog-AMS é o uso das variáveis *parameters*. Este tipo de variável fica disponível para modificação (atalho: Q) como qualquer outro componente existente no Cadence.

O ADE L configurado para simulação transiente encontra-se na sequência, nota-se que foram utilizados parâmetros para a quantidade de tempo da simulação, $VAR("Cycles")$, esta abordagem possibilita o controle exato da quantidade de períodos simulados. É possível salvar o estado da simulação em: *"Session -> Save State -> Cellview"*.

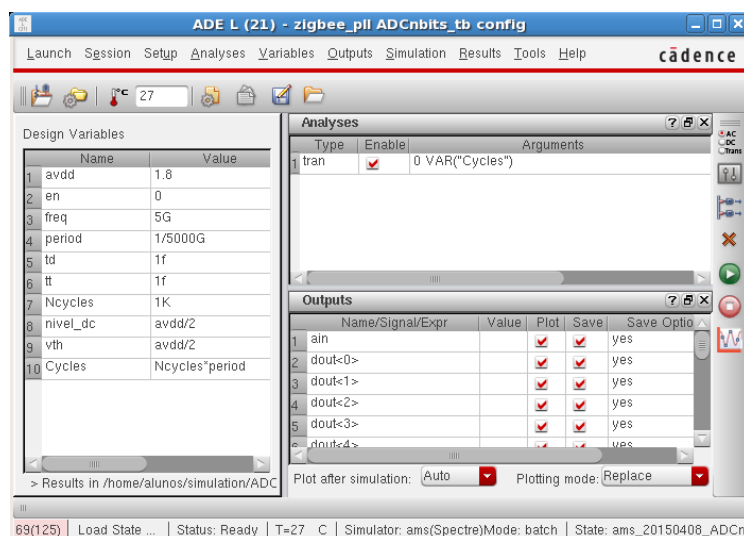


Figura 95 – ADE L para simulação do conversor AD modelado.

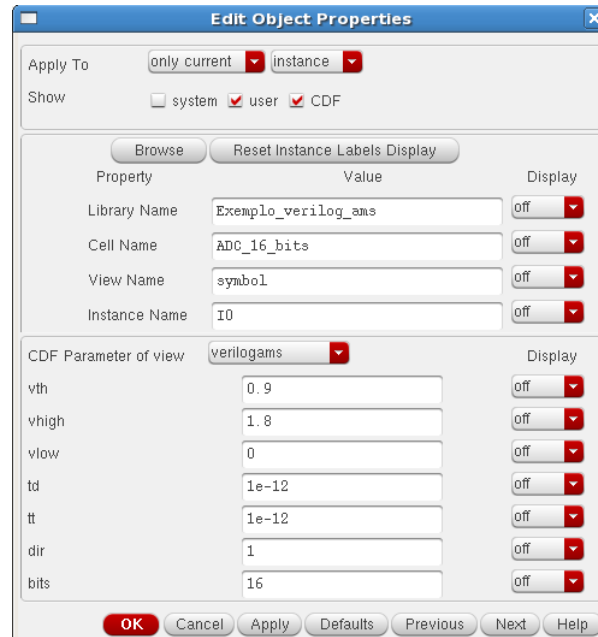


Figura 96 – Parâmetros disponíveis para o conversor AD modelado.

A simulação do conversor AD de 16 *bits* modelado encontra-se na sequência.

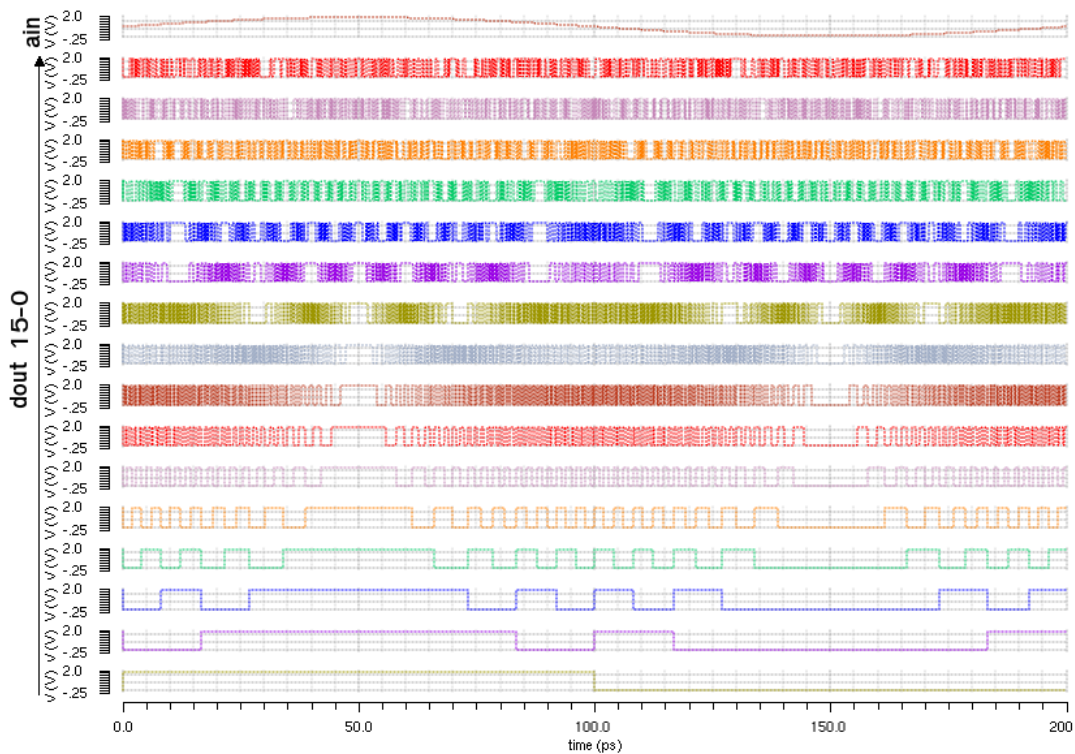


Figura 97 – Simulação do conversor AD de 16 *bits* modelado em Verilog-AMS.

Por fim, vale comentar sobre os comandos usados para retirar eventuais *locks* que travam os projetos. Para isto, abre-se uma nova aba do terminal em uso (*ctrl+shift+t*), digita-se, "*csh*" e "*clsAdminTool*", onde nesta ferramenta, digita-se "*are .*", desta forma, os *locks* são excluídos. Esta rotina é muito usada quando deseja-se mudar a *library* de pasta, onde modifica-se o arquivo *cds.lib*, incluindo a *library*, e retiram-se os *locks*.


```

%      - P: Programmable Counter(P);
%      - Wn: frequencia natural;
%      - Wc: largura de banda do loop;
%      - Parametros do Filtro de Malha - filtro passivo de segunda ordem
%          C1, C2 e R1
%      - T_L: (settling time) tempo de acomodacao;

clc; close all;
fprintf('=====');
fprintf('\n\n      Sintetizador de Frequencia PLL N-Inteiro para %s\n\n', Aplicacao);

%=====

%Imprimindo parametros de entrada
fprintf('===== ENTRADAS =====\n\n');
fprintf('Espacamento do Canal: %.2fMHz\n', Espacamento_Canal/10^6);
fprintf('Espectro: de %.2fMHz a %.2fMHz\n', Espectro(1)/10^6, ...
        Espectro(2)/10^6);
fprintf('Vctr: de %.2fV a %.2fV\n', Vctr(1), Vctr(2));
fprintf('Corrente do Charge pump (I_CP): %.2fuA\n', I_CP*10^6);
fprintf('Tempo de lock (T_L): %.2fus\n', T_L*10^6);

%=====

%Variaveis
V_referencia = 1.8;
Fsaida = [0 0];
Fref = Espacamento_Canal;

%=====

%Fsaida
%Teste no espectro
if Espectro(1) > Espectro(2)
    fprintf('Erro no espectro\n');
    quit cancel;
end

%Testando limites inferiores
Resto = mod(Espectro(1), Espacamento_Canal);
if Resto ~= 0
    Fsaida(1) = Espectro(1) + Resto;
else
    Fsaida(1) = Espectro(1);
end

%Testando limites superiores
Resto = 0;
Resto = mod(Espectro(2), Espacamento_Canal);
if Resto ~= 0
    Fsaida(2) = Espectro(2) - Resto;
else
    Fsaida(2) = Espectro(2);
end
Fsaida(2) = Fsaida(2) - Espacamento_Canal;

fprintf('\n===== Fsaida =====\n\n');
fprintf('Fsaida: de %.2fMHz a %.2fMHz\n', (Fsaida(1))/1000000, ...
        Fsaida(2)/1000000);

```

```

%=====

%Fvco
if Db_Fq_VCO_Flag == 0
    Fvco = [Fsaida(1) Fsaida(2)];
    Fvco_c = (Fsaida(1) + Fsaida(2));
else
    Fvco = [Fsaida(1)-50e6 Fsaida(2)+50e6];
    Fvco_c = (Fsaida(1) + Fsaida(2));
end

fprintf('\n===== VCO =====\n\n');
fprintf('Fvco: de %.2fMHz a %.2fMHz\n', Fvco(1)/1000000, ...
        Fvco(2)/1000000);
fprintf('Fvco_central: %.2fMHz\n', Fvco_c/2000000);

%=====

%Ganho do VCO
if Vctr(1) < 0 || Vctr(2) < 0 || Vctr(1) > V_referencia || Vctr(2) > ...
    V_referencia || Vctr(1) > Vctr(2)
    fprintf('Erro no Vctl\n');
    quit cancel;
else
    Kvco = (Fvco(2) - Fvco(1))/(Vctr(2) - Vctr(1));
    fprintf('Kvco: %.2fMHz/V\n', Kvco/1000000);
end

%=====

%N
N = (Fsaida(1) + Fsaida(2))/(Espacamento_Canal);

fprintf('\n===== Divisor =====\n\n');
fprintf('Divisor N-inteiro: de %d a %d\n\tDobro da Media: %d\n', ...
        (Fsaida(1))/Espacamento_Canal, ...
        Fsaida(2)/Espacamento_Canal, N);

%=====

%Parametros do divisor
S = (Fsaida(2) - (Fsaida(1) - Espacamento_Canal))/Espacamento_Canal;
M = S - 1;
P = ((Fsaida(1))/Espacamento_Canal)/M;

fprintf('Parametros do divisor   %d/%d\n\tM: %d ou %d\n\tS: %d\t\t%d bits\n\tP: ...
        %d\t\t%d bits\n\n', M, M+1, M, M+1, S, round(log2(S)), P, round(log2(P)));

%=====

%Ruido de Fase
% - PN: contribuicao do ruido de fase no Oscilador Local (LO);
% - PLO1 e PLO2: conteudo de energia do sinal.

fprintf('===== Ruido de Fase =====\n\n');
if strcmp(Aplicacao, 'ZigBee') == 1
    fprintf('(PN-P_LO)05MHz: %dBc/Hz\n', ...
            round((Psig1 - Pint1) - 10*log10(BW) - SNRmin - tol1));
    fprintf('(PN-P_LO)10MHz: %dBc/Hz\n\n', ...
            round((Psig2 - Pint2) - 10*log10(BW) - SNRmin - tol1));

```

```

else
    fprintf(' (PN-P_LO)05MHz: %ddBc/Hz\n', ...
        round((Psig1 - Pint1) - 10*log10(BW) - SNRmin - tol1));
    fprintf(' (PN-P_LO)10MHz: %ddBc/Hz\n\n', ...
        round((Psig2 - Pint2) - 10*log10(BW) - SNRmin - tol1));
end

%=====
%Rejeicao de Espurios
% - Pspurs: conteudo de energia dos espurios;
% - PLO1 e PLO2: conteudo de energia do sinal.

fprintf('===== Rejeicao de Espurios =====\n\n');
if strcmp(Aplicacao, 'ZigBee') == 1
    fprintf(' (P_Spurs-P_LO)05MHz: %ddBc/Hz\n', ...
        (Psig1 - Pint1) - SNRmin - tol1);
    fprintf(' (P_Spurs-P_LO)10MHz: %ddBc/Hz\n\n', ...
        (Psig2 - Pint2) - SNRmin - tol1);
else
    fprintf(' (P_Spurs-P_LO)05MHz: %ddBc/Hz\n', ...
        (Psig1 - Pint1) - SNRmin - tol1);
    fprintf(' (P_Spurs-P_LO)10MHz: %ddBc/Hz\n\n', ...
        (Psig2 - Pint2) - SNRmin - tol1);
end

%=====
%Filtro de Malha
% Wn: frequencia natural;
% Wc: largura de banda do loop;
% E: fator de amortecimento, comumente escolhido 0.707 ou 1;
% Wz1 e Wp1: localizacao dos polos e zeros para amortecimento critico;
% PM: Margem de Fase.

Wn_min = 2*pi/T_L;
Wn_max = 2*pi*Fref/10;
while_cond = 0;

fprintf('===== Parametros do PLL =====\n\n');
fprintf('Fator de amortecimento usado: %d\n', Fator_Amortecimento);
fprintf('Frequencia natural: %.2fKrad/s < Wn < %.2fMrad/s\n', ...
    Wn_min/10^3, Wn_max/10^6);
Wn_str = input('Escolha uma frequencia natural no intervalo (rad/s): ', 's');
Wn = sscanf(Wn_str, '%f');
fprintf('Wn escolhido: %.2fKrad/s\t\t %.2fKHz\n', ...
    Wn/10^3, Wn/(2*pi*10^3));

if Wn > Wn_min && Wn < Wn_max
    while_cond = 1;
end

while while_cond == 0
    Wn_str = input('\n\nWn invalido\nEscolha uma frequencia natural
        ...no intervalo (rad/s): ', 's');
    Wn = sscanf(Wn_str, '%f');
    fprintf('Wn escolhido: %.2fKrad/s\t\t %.2fKHz\n', ...
        Wn/10^3, Wn/(2*pi*10^3));

    if Wn > Wn_min && Wn < Wn_max

```



```

        while_cond = 1;
    end
end

Wc = 2*Fator_Amortecimento*Wn;
Wz1 = Wc/(2^2);
Wp1 = Wc*(2^2);
PM = atan(Wc/Wz1) - atan(Wc/Wp1);

fprintf('Largura de banda Wc: %.2fKrad/s\n', Wc/1000);
fprintf('Margem de fase: %.2f graus\n\n', 57.2957795*PM);

%=====

%Componentes do filtro
C1 = I_CP*(2*pi*Kvco)/(2*pi*N*(Wn)^2);
R1 = 1/(Wz1*C1);
C2 = 1/(Wp1*R1);

fprintf('===== Parametros do Filtro =====\n\n');
fprintf('C1: %.2fF\nC2: %.2fF\nR1: %.2fOhm\n', ...
        C1*10^12, C2*10^12, R1/1000);

%=====

%Settling time
ln = @(x)(log(x));
t_lock = (1/(Fator_Amortecimento*Wn))*ln((Fvco(2) - ...
        (Fvco(1) - 2*Espacamento_Canal))/((Acuracia_Frequencia/10^6)*Fvco_c));
fprintf('tlock: %.2fus %.2f%% de %.2fus\n\n', ...
        t_lock*10^6, (t_lock/T_L)*100, T_L*10^6);

%=====

%Graficos
%Ganho em Malha Aberta
figure;
k1 = (2*pi*Kvco)*I_CP*Wp1/(2*pi*C1*N*Wz1);
num1 = [k1 k1*Wz1];
den1 = [1 Wp1 0 0];
sys1 = tf(num1, den1);
bode(sys1); grid on;
title('Ganho em Malha Aberta');
[~,Pm,Wgm,Wpm] = margin(sys1);

%Ganho em Malha Fechada
figure;
num2 = [Wn^2];
den2 = [1 2*Fator_Amortecimento*Wn Wn^2];
sys2 = tf(num2, den2);
bode(sys2); grid on;
title('Ganho em Malha Fechada');
BW_CL = bandwidth(sys2);

fprintf('===== Parametros dos Graficos =====\n\n');
fprintf('Malha Aberta: \n\tMargem de Fase: %.2fgraus\n', Pm);
fprintf('Malha Fechada:\n\tLargura de Banda: %.2fKrad/s ou %.2fKHz\n\n', ...
        BW_CL/10^3, BW_CL/(2*pi*10^3));

%=====

```

```

%Simulacao para o protocolo ZigBee
if strcmp(Aplicacao, 'ZigBee') == 1
    quit cancel;

%Simulacoes em Malha Aberta
fprintf('===== Simulacoes em Malha Aberta =====\n\n');
Fator_Amortecimento = 1;

%MIN
figure;
Wn = Wn_min;
Wc = 2*Fator_Amortecimento*Wn;
Wz1 = Wc/(2^2);
Wp1 = Wc*(2^2);

%Componentes do filtro de malha
C1 = I_CP*(2*pi*Kvco)/(2*pi*N*(Wn)^2);
R1 = 1/(Wz1*C1);
C2 = 1/(Wp1*R1);

%figura;
k1 = (2*pi*Kvco)*I_CP*Wp1/(2*pi*C1*N*Wz1);
num1 = [k1 k1*Wz1];
den1 = [1 Wp1 0 0];
sys1 = tf(num1, den1);
bode(sys1); grid on;
title('Ganho em Malha Aberta');
hold on;

%INTERMEDIARIO 1
for Wn = 40e3: 10e3: 90e3
    Wc = 2*Fator_Amortecimento*Wn;
    Wz1 = Wc/(2^2);
    Wp1 = Wc*(2^2);

    %Componentes do filtro de malha
    C1 = I_CP*(2*pi*Kvco)/(2*pi*N*(Wn)^2);
    R1 = 1/(Wz1*C1);
    C2 = 1/(Wp1*R1);

    k1 = (2*pi*Kvco)*I_CP*Wp1/(2*pi*C1*N*Wz1);
    num1 = [k1 k1*Wz1];
    den1 = [1 Wp1 0 0];
    sys1 = tf(num1, den1);
    bode(sys1);
    hold on;

end

%INTERMEDIARIO 2
for Wn = 100e3: 100e3: 400e3
    Wc = 2*Fator_Amortecimento*Wn;
    Wz1 = Wc/(2^2);
    Wp1 = Wc*(2^2);

    %Componentes do filtro de malha
    C1 = I_CP*(2*pi*Kvco)/(2*pi*N*(Wn)^2);
    R1 = 1/(Wz1*C1);
    C2 = 1/(Wp1*R1);

```

```

k1 = (2*pi*Kvco)*I_CP*Wp1/(2*pi*C1*N*Wz1);
num1 = [k1 k1*Wz1];
den1 = [1 Wp1 0 0];
sys1 = tf(num1, den1);
bode(sys1); grid on;
title('Ganho em Malha Aberta');
hold on;

end

%INTERMEDIARIO 3
for Wn = 500e3: 500e3: 3e6
    Wc = 2*Fator_Amortecimento*Wn;
    Wz1 = Wc/(2^2);
    Wp1 = Wc*(2^2);

    %Componentes do filtro de malha
    C1 = I_CP*(2*pi*Kvco)/(2*pi*N*(Wn)^2);
    R1 = 1/(Wz1*C1);
    C2 = 1/(Wp1*R1);

    k1 = (2*pi*Kvco)*I_CP*Wp1/(2*pi*C1*N*Wz1);
    num1 = [k1 k1*Wz1];
    den1 = [1 Wp1 0 0];
    sys1 = tf(num1, den1);
    bode(sys1);
    hold on;

end

%MAX
Wn = Wn_max;
Wc = 2*Fator_Amortecimento*Wn;
Wz1 = Wc/(2^2);
Wp1 = Wc*(2^2);

%Componentes do filtro de malha
C1 = I_CP*(2*pi*Kvco)/(2*pi*N*(Wn)^2);
R1 = 1/(Wz1*C1);
C2 = 1/(Wp1*R1);

k1 = (2*pi*Kvco)*I_CP*Wp1/(2*pi*C1*N*Wz1);
num1 = [k1 k1*Wz1];
den1 = [1 Wp1 0 0];
sys1 = tf(num1, den1);
bode(sys1);
hold on;

%=====

%Simulacoes em Malha Fechada
fprintf('===== Simulacoes em Malha Fechada =====\n\n');

%MIN
Wn = Wn_min;
Wc = 2*Fator_Amortecimento*Wn;
Wz1 = Wc/(2^2);
Wp1 = Wc*(2^2);

```

```

%Componentes do filtro de malha
C1 = I_CP*(2*pi*Kvco)/(2*pi*N*(Wn)^2);
R1 = 1/(Wz1*C1);
C2 = 1/(Wp1*R1);

figure;
num2 = [Wn^2];
den2 = [1 2*Fator_Amortecimento*Wn Wn^2];
sys2 = tf(num2, den2);
bode(sys2); grid on;
title('Ganho em Malha Fechada');
BW_CL = bandwidth(sys2);
hold on;

%INTERMEDIARIO 1
for Wn = 40e3: 10e3: 90e3
    Wc = 2*Fator_Amortecimento*Wn;
    Wz1 = Wc/(2^2);
    Wp1 = Wc*(2^2);

    %Componentes do filtro de malha
    C1 = I_CP*(2*pi*Kvco)/(2*pi*N*(Wn)^2);
    R1 = 1/(Wz1*C1);
    C2 = 1/(Wp1*R1);

    num2 = [Wn^2];
    den2 = [1 2*Fator_Amortecimento*Wn Wn^2];
    sys2 = tf(num2, den2);
    bode(sys2);
    hold on;

end

%INTERMEDIARIO 2
for Wn = 100e3: 100e3: 400e3
    Wc = 2*Fator_Amortecimento*Wn;
    Wz1 = Wc/(2^2);
    Wp1 = Wc*(2^2);

    %Componentes do filtro de malha
    C1 = I_CP*(2*pi*Kvco)/(2*pi*N*(Wn)^2);
    R1 = 1/(Wz1*C1);
    C2 = 1/(Wp1*R1);

    num2 = [Wn^2];
    den2 = [1 2*Fator_Amortecimento*Wn Wn^2];
    sys2 = tf(num2, den2);
    bode(sys2);
    hold on;

end

%INTERMEDIARIO 3
for Wn = 500e3: 500e3: 3e6
    Wc = 2*Fator_Amortecimento*Wn;
    Wz1 = Wc/(2^2);
    Wp1 = Wc*(2^2);

```

```

%Componentes do filtro de malha
C1 = I_CP*(2*pi*Kvco)/(2*pi*N*(Wn)^2);
R1 = 1/(Wz1*C1);
C2 = 1/(Wp1*R1);

num2 = [Wn^2];
den2 = [1 2*Fator_Amortecimento*Wn Wn^2];
sys2 = tf(num2, den2);
bode(sys2);
hold on;

end

%MAX
Wn = Wn_max;
Wc = 2*Fator_Amortecimento*Wn;
Wz1 = Wc/(2^2);
Wp1 = Wc*(2^2);

%Componentes do filtro de malha
C1 = I_CP*(2*pi*Kvco)/(2*pi*N*(Wn)^2);
R1 = 1/(Wz1*C1);
C2 = 1/(Wp1*R1);

%figura
num2 = [Wn^2];
den2 = [1 2*Fator_Amortecimento*Wn Wn^2];
sys2 = tf(num2, den2);
bode(sys2);
hold on;

%=====

%Filtro de Malha parameters simulation
fprintf('===== Filtro de Malha Parameters Simulations =====\n\n');
k = 1;

for Wn = Wn_min: 10e3: Wn_max
    Wc = 2*Fator_Amortecimento*Wn;
    Wz1 = Wc/(2^2);
    Wp1 = Wc*(2^2);

    %Componentes do filtro de malha
    C1 = I_CP*(2*pi*Kvco)/(2*pi*N*(Wn)^2);
    R1 = 1/(Wz1*C1);
    C2 = 1/(Wp1*R1);

    C_1(k) = C1;
    C_2(k) = C2;
    R_1(k) = R1;
    Wn_1(k) = Wn;
    Wc_1(k) = Wc;
    k = k + 1;

end

figure;
plot(Wn_1/(2*pi*10^3), C_1*10^12, 'g');
hold on;
plot(Wn_1/(2*pi*10^3), C_2*10^12, 'r');

```

```

    hold on;
    plot(Wn_1/(2*pi*10^3), R_1/(10^3), 'b');
    grid on;
    title('Relacao entre C1, C2 e R2 com fn');
    ylabel('C1[pF] C2[pF] R1[KOhms]');
    xlabel('fn [KHz]');
    axis([0 500 0 3000]);
end

%=====

%Gerando valores para tabela - Parametros do Filtro de Malha
for Wn = 100e3: 20e3: 900e3
    Wc = 2*Fator_Amortecimento*Wn;
    Wz1 = Wc/(2^2);
    Wp1 = Wc*(2^2);

    fprintf('Wn: %.2fKrad/s ou %.2fKHz\tWc: %.2fKrad/s ou %.2fKHz\n', ...
        Wn/10^3, Wn/(2*pi*10^3), Wc/10^3, Wc/(2*pi*10^3));

    %Componentes do filtro de malha
    C1 = I_CP*(2*pi*Kvco)/(2*pi*N*(Wn)^2);
    R1 = 1/(Wz1*C1);
    C2 = 1/(Wp1*R1);

    fprintf('C1: %.2fpF\tC2: %.2fpF\tR1: %.2fkOhms\n\n', ...
        C1*10^12, C2*10^12, R1/10^3);

end

fprintf('===== \n\n');

end

```

```

=====

      Sintetizador de Frequência PLL N-Inteiro para ZigBee

===== ENTRADAS =====

Espacamento do Canal: 5.00MHz
Espectro: de 2400.00MHz a 2483.50MHz
Vctr: de 0.50V a 1.50V
Corrente do Charge pump (I_CP): 20.00uA
Tempo de lock (T_L): 192.00us

===== Fsaida =====

Fsaida: de 2400.00MHz a 2475.00MHz

===== VCO =====

Fvco: de 2400.00MHz a 2475.00MHz
Fvco_central: 2437.50MHz
Kvco: 75.00MHz/V

===== Divisor =====

Divisor N-inteiro: de 480 a 495
      Dobro da Média: 975
Parâmetros do divisor   15/16
      M: 15 ou 16
      S: 16           4 bits
      P: 32           5 bits

===== Ruído de Fase =====

(PN-P_L0)05MHz: -80dBc/Hz
(PN-P_L0)10MHz: -110dBc/Hz

===== Rejeição de Espúrios =====

(P_Spurs-P_L0)05MHz: -13dBc/Hz
(P_Spurs-P_L0)10MHz: -43dBc/Hz

===== Parâmetros do PLL =====

Fator de amortecimento usado: 1
Frequência natural: 32.72Krad/s < Wn < 3.14Mrad/s
Escolha uma frequência natural no intervalo (rad/s): 310100
Wn escolhido: 310.10Krad/s           49.35KHz
Largura de banda Wc: 620.20Krad/s
Margem de fase: 61.93 graus

===== Parâmetros do Filtro =====

C1: 16.00pF
C2: 1.00pF
R1: 403.13kOhm
tlock: 19.60us 10.21% de 192.00us

===== Parâmetros dos Gráficos =====

Malha Aberta:
      Margem de Fase: 61.93graus
Malha Fechada:
      Largura de Banda: 199.17Krad/s ou 31.70KHz

```

Figura 98 – Resultados obtidos do código D.1.

```

=====

    Sintetizador de Frequência PLL N-Inteiro para ZigBee

===== ENTRADAS =====

Espacamento do Canal: 5.00MHz
Espectro: de 2400.00MHz a 2483.50MHz
Vctr: de 0.50V a 1.50V
Corrente do Charge pump (I_CP): 20.00uA
Tempo de lock (T_L): 192.00us

===== Fsaída =====

Fsaída: de 2400.00MHz a 2475.00MHz

===== VCO =====

Fvco: de 2350.00MHz a 2525.00MHz
Fvco_central: 2437.50MHz
Kvco: 175.00MHz/V

===== Divisor =====

Divisor N-inteiro: de 480 a 495
    Dobro da Média: 975
Parâmetros do divisor    15/16
    M: 15 ou 16
    S: 16          4 bits
    P: 32          5 bits

===== Ruído de Fase =====

(PN-P_LO)05MHz: -80dBc/Hz
(PN-P_LO)10MHz: -110dBc/Hz

===== Rejeição de Espúrios =====

(P_Spurs-P_LO)05MHz: -13dBc/Hz
(P_Spurs-P_LO)10MHz: -43dBc/Hz

===== Parâmetros do PLL =====

Fator de amortecimento usado: 1
Frequência natural: 32.72Krad/s < Wn < 3.14Mrad/s
Escolha uma frequência natural no intervalo (rad/s): 473700
Wn escolhido: 473.70Krad/s          75.39KHz
Largura de banda Wc: 947.40Krad/s
Margem de fase: 61.93 graus

===== Parâmetros do Filtro =====

C1: 16.00pF
C2: 1.00pF
R1: 263.92kOhm
tlock: 14.47us 7.54% de 192.00us

===== Parâmetros dos Gráficos =====

Malha Aberta:
    Margem de Fase: 61.93graus
Malha Fechada:
    Largura de Banda: 304.25Krad/s ou 48.42KHz

```

Figura 99 – Resultados obtidos do código D.1 com modificação do ganho do VCO.

Anexos

ANEXO A – Fonte de Corrente

A fonte de corrente usada no projeto do VCO é a apresentada a seguir. Trata-se de uma fonte de corrente com compensação de temperatura, onde apresenta a geração de um nível DC de corrente com baixa variação frente a oscilação de temperatura. A faculdade UnB Gama trabalha atualmente na construção de uma biblioteca de IPs, contendo blocos funcionais para a reutilização em projetos de circuitos integrados complexos, sendo este bloco componente desta biblioteca e adequando-se perfeitamente a este projeto. O esquemático, *testbench* e simulação estão dispostos na sequência.

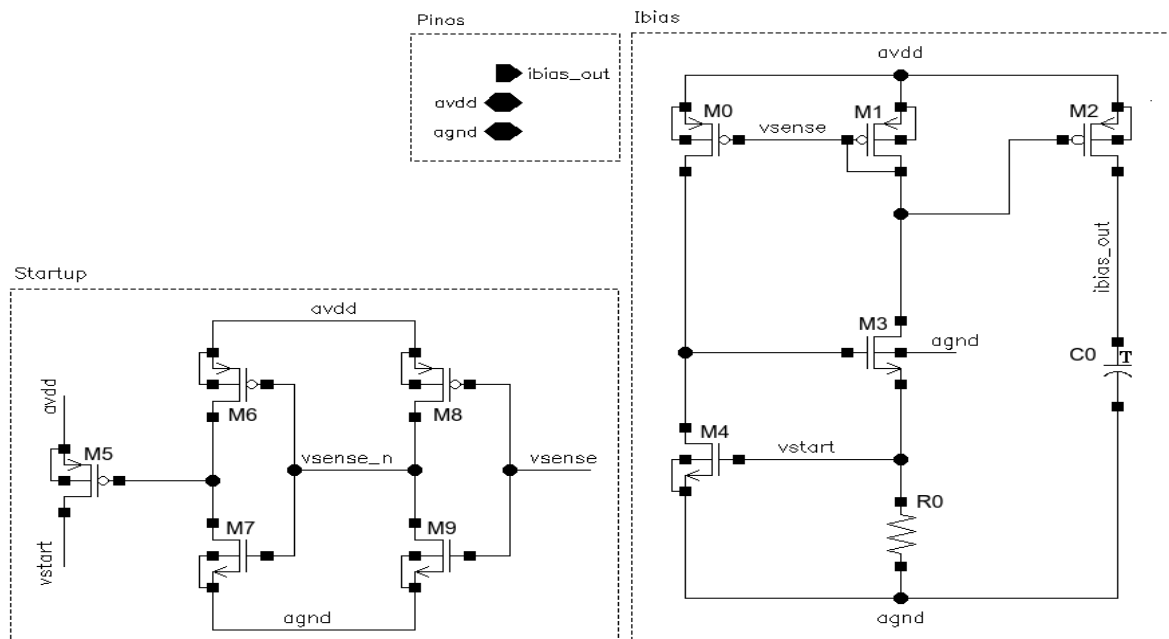


Figura 100 – Esquemático do fonte de corrente usada no VCO.

Os componentes utilizados e valores dos mesmos são:

- M0 e M1: $(W/L) = 20\mu\text{m} / 2\mu\text{m}$, $fingers = 1$, $simM = 1$, $totalM = 1$;
- M2: $(W/L) = 10\mu\text{m} / 2\mu\text{m}$, $fingers = 1$, $simM = 1$, $totalM = 1$;
- M3 e M4: $(W/L) = 50\mu\text{m} / 2\mu\text{m}$, $fingers = 1$, $simM = 1$, $totalM = 1$;
- M5: $(W/L) = 5\mu\text{m} / 10\mu\text{m}$, $fingers = 1$, $simM = 1$, $totalM = 1$;
- M6 e M7: $(W/L) = 1\mu\text{m} / 1\mu\text{m}$, $fingers = 1$, $simM = 1$, $totalM = 1$;
- M8 e M9: $(W/L) = 1\mu\text{m} / 1\mu\text{m}$, $fingers = 1$, $simM = 1$, $totalM = 1$;
- C0 (*mimcap_2p0_sin*): $c = 35.6\text{fF}$, $l = 4\mu\text{m}$, $w = 4\mu\text{m}$, $mLv = 6$, $m = 1$;
- R0 (*rnpo 1rpo*): $res = 46.2693\text{K}\Omega$, $sumW = 2\mu\text{m}$, $sumL = 3.000.000\mu\text{m}$, $m = 1$.

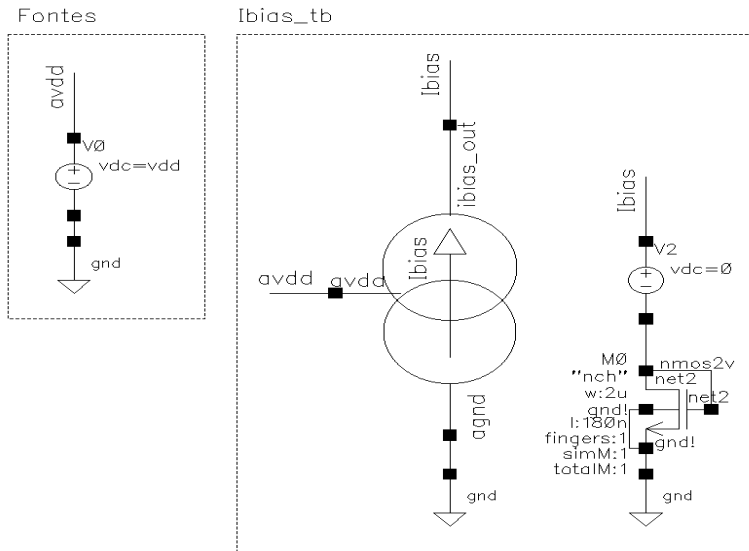


Figura 101 – *Testebench* da fonte de corrente usada no VCO.

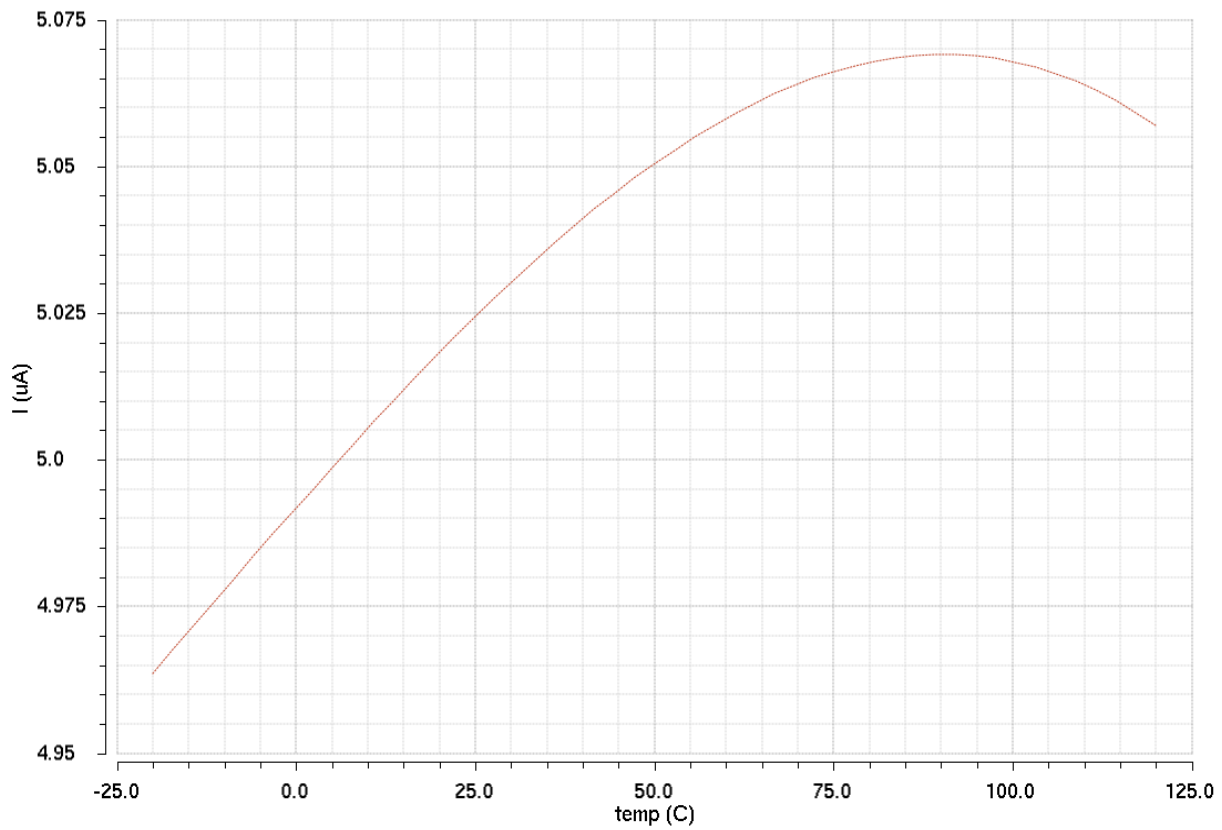


Figura 102 – Simulação da fonte de corrente usada no VCO.